3D Mapping of an Unknown Environment with a Robust Cooperation Team of Mobile Robots

Zur Erlangung des akademischen Grades eines

Doktors der Natur-/Ingenieurswissenschaften

der Fakultät für Informatik

der Universität Karlsruhe (Technische Hochschule)

genehmigte

Dissertation

von

Ashraf S. S. Huwedi

aus Benghazi - Libya

Tag der mündlichen Prüfung:	15.07.2008
Erster Gutachter:	Prof. Dr.-Ing. R. Dillmann
Zweiter Gutachter:	Prof. Dr.-Ing. H. Wörn

Contents

Abstract

Some of the important future applications of mobile robot systems are autonomous indoor and outdoor measurements of building, factories and objects in three dimensional view. A goal of such a measurement is to provide us with a detailed map of the environment with the interesting characteristics. This map can then be transferred into a model, which represents the measuring objects. By following an automated proceeding, an autonomous measurement robot could be useful in order to extract independently the map and model of the environment. The environmental model can then be directly used by the autonomous mobile robot for navigation. A further increase of the effectiveness in map production can be achieved, if the environment is mapped by several robots, or whole robot fleets. Such an approach can be advantage, for example, by reducing the exploration duration.

Most of the multi robot systems applied for mapping do not use explicit cooperation and communication to avoid the complexity of the cooperation aspects. The simple solution would be in these non-cooperative cases, to let the robots explore different areas. These areas are divided between them in relation to their mapping speeds to make them finish their tasks at the same time. Each robot has to solve its part of the task without knowing any information about the status of the other robots. Such a partitioning of the areas sounds to be an optimal solution, but this is only the case if the robots are not to be robustly cooperative. So this way of splitting the task among the robots is not as efficient as it could be, and could be improved if the robots cooperate and help each other while doing their tasks. To be able to do so, all robots have to exchange information in order to achieve the tasks very quickly, and to coordinate themselves over the environment efficiently.

Coordination, which is an essential characteristic of any task-achieving multi-robot system, should be addressed in order to explore the environment efficiently without any conflicts or sensor interference. Within this case, a framework architecture is needed to be designed in order to allow the development of multi-robot system that can manage the coordination problem efficiently. The responsibilities of this framework are also to achieve an exploration strategy, which allows different tasks to be assigned to each robot in order to implement the tasks as efficiently as possible.

This thesis develops the framework for 3D mapping using multi-mobile robots. Two mobile robots equipped with different sensors capabilities are used. The main contribution of this thesis is to autonomously build a 3D map of indoor environments within a good exploration time by using multi-mobile robots. During the process, the exploration algorithm should choose appropriate motion commands to let the robots achieve the tasks as quickly as possible. A good collaboration among a team of robots should be achieved in order to let them efficiently solve the exploration task. As a result, the overall time to complete the exploration mission should be reduced. The coordination among the robots should be taken into consideration so that they efficiently distribute themselves over the environment avoiding redundant work and reducing the risk of interference between the vehicles.

In order to achieve such goal efficiently, some methodologies and approaches are developed within the course of this work. Firstly, a framework for 3D mapping using multi-mobile robots is designed. A master-slave approach is suggested in order to avoid redundant work and to coordinate among the robots. This approach allows the robots to accomplish a tightly coupled exploration of the environment. The communication module within this structure allows for robot-to-robot communication using different types of protocols, in order to let the robots cooperate in an efficient manner. The framework also deals with the problem of coordination of robots to avoid collision in situations where two robots are performing different tasks close to each other.

Further, a motion command algorithm is developed, in order to distribute the robots over the environment. The main objective of this algorithm is to let the robots move to a new unexplored place, based on certain prospect values of this place, trying to avoid any work conflicts or sensor interference problem. The natural feature quality available at that place, the navigation cost to it and the coordination factor are three prospects, which are used to calculate the total prospect of each target in order to choose the best one of several interest places. The algorithm is tested using different approaches: using one master, using two masters or using one master and n slaves.

Furthermore, different cooperative strategies among the robots are implemented. To study the efficiency of the prospect values designed in the motion control algorithm, a simulation package is implemented in order to study in more details the performance of the system applying such prospect values. Three strategies are implemented in order to evaluate the performance of the system by using two masters, whereas two strategies are implemented to evaluate the system performance by using n slaves.

Finally, a feature extraction algorithm is developed. Natural landmarks available in the indoor environment have to be identified based on some predefined knowledge about their features, as for example the texture behaviour or geometrical structure. This algorithm is developed in a way that the extraction of geometric features can be obtained by using different sensor capabilities. A segmentation process is developed and implemented by using region growing techniques, which allows the algorithm to be used online. Some classification rules are applied to identify the located features.

This thesis also investigates the performance evaluation of a master-slave system developed within the course of this work. The evaluation is achieved considering three main evaluation criteria, which are concerning of the time of exploration, of the coordination efficiency and of 3D mapping of a complex environment.

The performance of the developed master-slave system is evaluated in chapter 7. It is noticed that the use of multiple robots results in a better exploration time than the use of a single one. The exploration time of all strategies is considerably improved compared to the case of only one robot by more than one half of its exploration time. Furthermore within the case of multiple robots, a dynamic strategy in case of master structure and distributed strategy in case of slave structure are found to be the best strategy, which have a good exploration time.

Concerning the coordination efficiency, good results are obtained by developing some cooperative strategies that enables n slaves to coordinate themselves over the environment and

to distribute tasks among them in an efficient way. As a result with strong coordination, the time of the exploration is improved and the system performance is in general stable.

A 3D digitalized model of a complex indoor environment is built by using two mobile robots equipped with different sensor capabilities. The experiments are carried out in the first floor of the computer center building. Three different forms of 3D model are demonstrated such as in the form of point clouds, in the form of triangulated points and in the form of the same building with cutting the ceiling of it. As a general conclusion, the 3D map is obtained in a way that the map is closed in an efficient way.

Zusammenfassung

Lösungsansatz und Beitrag der Arbeit

Einige der wichtigsten zukünftigen Anwendungen mobiler Robotersysteme sind autonome, dreidimensionale Innen- und Außenvermessungen von Gebäuden, Fabrikanlagen und Gegenständen. Eines der Ziele solcher Messungen ist es, einen möglichst präzisen und detaillierten Umgebungsplan zu liefern, welche alle für die gegebene Aufgabe relevanten Merkmalen der Umgebung enthält. Dieser Plan kann dann in ein Modell des zu vermessenden Objektes umgewandelt werden. Bei einer solchen automatischen Vorgehensweise kann ein autonomer Messroboter benutzt werden, um den Umgebungsplan und das Umgebungsmodell selbstständig zu erstellen. Das Umgebungsmodell kann dann direkt von einem autonomen mobilen Roboter für die Navigation benutzt werden. Eine bessere Effektivität bei der Planherleitung kann erreicht werden, wenn die Umgebung durch mehrere Roboter abgebildet wird. Ein solcher Ansatz ist beispielsweise dann vorteilhaft, wenn hierdurch die Explorationsdauer reduziert wird.

Die meisten in der Kartierung eingesetzten Multirobotersysteme benutzen keine explizite Kooperation oder Kommunikation, um die Komplexität des Systems möglichst gering zu halten. Eine im nicht-kooperativen Fall verwendete einfache Lösung wäre es, die Roboter unterschiedliche Bereiche erforschen zu lassen. Diese Bereiche werden in Abhängigkeit von der Kartierungsgeschwindigkeit des jeweiligen Roboters bestimmt, sodass alle Roboter ihre Aufgabe zum selben Zeitpunkt beenden. Hierbei muss jeder Roboter in der Lage sein, seine Teilaufgabe zu erledigen, ohne zu wissen, wie weit die anderen Roboter mit ihren jeweiligen Teilaufgaben fortgeschritten sind. Eine solche Flächenpartitionierung scheint zwar zunächst eine optimale Lösung zu sein, dies trifft aber nur im Falle einer nicht robusten Kooperation zu. Somit ist dies nicht die effizienteste Art der Aufgabenverteilung zwischen Robotern. Sie kann verbessert werden, indem die Roboter miteinander kooperieren und sich gegenseitig beim Ausführen der Teilaufgaben helfen. Die Roboter müssen also Informationen austauschen, um sich effizient auf der zu erforschenden Fläche verteilen zu können.

Die Koordination der Roboter als eine wesentliche Charakteristik eines jeden Multirobotersystems musste im Rahmen dieser Arbeit gelöst werden, um die Umgebung ohne Kollision oder Sensorinterferenz effizient zu erforschen. Hierbei sollte eine Rahmenarchitektur entworfen werden, die die Entwicklung eines Multirobotersystems zur effizienten Lösung des Koordinationsproblems erlaubt. Ziel war es dabei, eine Erforschungsstrategie zu entwickeln, bei der jedem Roboter unterschiedliche Teilaufgaben übertragen werden können, um die gegebene Aufgabe so effizient wie möglich auszuführen.

Das Hauptziel dieser Dissertation ist die Erstellung einer 3D-Karte von Innenumgebungen innerhalb einer guten Explorationszeit unter Verwendung von mobilen Multirobotersystemen. Während dieses Prozesses wurden im Explorationsalgorithmus adäquate Bewegungskommandos so ausgewählt, dass die Roboter möglichst schnell ihre Aufgaben durchführen konnten. Eine gute Zusammenarbeit zwischen den Robotern war daher notwendig, um die Aufgabe effizient zu erledigen. Durch dieses Vorgehen konnte die Ausführungszeit für die

globale Explorationsaufgabe reduziert werden. Die Koordination wurde so angelegt, dass die Roboter sich optimal in der Umgebung verteilen und dabei gleichzeitig redundante Arbeitsschritte und das Risiko von Interferenzen vermeiden.

Um das Hauptziel erreichen zu können, wurden zwei mobile Roboter mit verschiedenen Sensorentypen ausgestattet. Dies wurde benötigt um autonome Vermessung und Kartierung von unbekanntem Gelände zu ermöglichen. Hierbei wurden insbesondere folgende Ansätze und Methodik entwickelt und untersucht:

- Ein Framework für eine 3D-Kartierung unter Verwendung eines mobilen Multirobotersystems wurde konzipiert. Hierfür wurde ein Master-Slave-Ansatz empfohlen, um redundante Aufgaben zu vermeiden und die Roboter zu koordinieren. Dieser Ansatz ermöglicht den Robotern eine eng gekoppelte Erforschung der Umgebung. In der entwickelten Architektur erlaubt das Datenübertragungsmodul eine Roboter-zu-Roboter-Kommunikation unter Verwendung verschiedener Protokolltypen. Dieses Framework befasst sich auch mit dem Problem der Koordination der Roboter, um bei räumlich nahe gelagerten Teilaufgaben Kollisionen zwischen Robotern zu vermeiden.

- Ein Algorithmus zur Bewegungssteuerung wurde entwickelt, um die Roboter in der Umgebung zu verteilen. Das wesentliche Ziel dabei war es, die Roboter basierend auf bestimmten Erwartungswerten von Umgebungsbereichen zu unerforschten Bereichen zu führen, wobei Arbeitskonflikte zwischen den Robotern und Sensorinterferenzen vermieden werden sollten. Die drei Erwartungswerte, welche verwendet wurden, um den globalen Erwartungswert einer Zielregion zu berechnen und die beste von mehreren möglichen Teilregionen auszuwählen, sind die Qualität der an diesem Ort verfügbaren relevanten Merkmalen, der Navigationsaufwand zu diesem Ort und ein Koordinationsfaktor. Der Algorithmus wurde unter Verwendung unterschiedlicher Ansätze getestet: mit einem Master, mit zwei Mastern und mit einem Master und n Slaves.

- Unterschiedliche Kooperationsstrategien wurden implementiert. Um die Effizienz der im Bewegungssteuerungsalgorithmus entworfenen Erwartungswerte zu untersuchen, wurde ein Simulationspaket implementiert, welches die detaillierte Untersuchung der Systemleistung Bei Anwendung dieser Erwartungswerte ermöglicht. Für den Einsatz von zwei Mastern wurden drei dieser Strategien implementiert: die Frontier-, die Subareas- und die Dynamics-Strategie. Die Evaluation der Systemperformanz beim n-Slaves-Ansatz erfolgte mit der Centralized- und der Distributed-Strategie.

- Weiterhin wurde ein Algorithmus zur Extraktion von Merkmalen entwickelt. Natürliche Landmarken im Innenbereich sollten identifiziert werden basierend auf Vorwissen über deren Merkmalen, wie beispielsweise ihr Texturverhalten oder ihre geometrische Struktur. Dieser Algorithmus wurde so konzipiert, dass die Extraktion geometrischer Merkmale unter Verwendung unterschiedlicher Sensorenprinzipien erfolgt. Ein Segmentierungsprozess wurde unter Verwendung des Region-growing-Prinzips so entwickelt und implementiert, dass er online verwendet werden konnte. Die Identifikation der lokalisierten Merkmale erfolgte mittels Klassifikationsregeln.

Evaluation des Gesamtssystems und Bewertung

Ziel dieser Arbeit ist der Entwurf und die Implementierung eines Systems zur Erstellung von 3D-Karten von Innenumgebungen mit mobilen Multirobotersystemen, wobei ein besonderes Augenmerk auf der Effizienz und der notwendigen Explorationszeit liegt. Das gegebene Ziel konnte unter Berücksichtigung der drei wesentlichen Evaluationskriterien erreicht werden: der Explorationsdauer, der Koordinationseffizienz und der 3D-Kartierung einer komplexen Umgebung.

In der Evaluierungsphase wurde die Leistung des entwickelten Master-Slave-Systems bewertet. Hierbei konnte festgestellt werden, dass der Einsatz von mehreren Robotern in einer verbesserten Explorationszeit resultiert im Vergleich zum Einsatz eines einzigen Roboters: Alle Multiroboterstrategien reduzieren die Explorationsdauer um deutlich mehr als die Hälfte im Vergleich zur Exploration mit einem einzelnen Roboter. Im Vergleich zwischen den unterschiedlichen Multiroboterstrategien erwiesen sich die Dynamic-Strategie in der Master-Struktur und die Distributed-Strategie in der Slave-Struktur als die besten Strategien, da sie die geringste Explorationsdauer benötigen.

Gute Ergebnisse bei der Koordinationseffizienz konnten erreicht werden durch die Entwicklung von Kooperationsstrategien, bei denen n Slaves in der Lage waren, sich optimal in der Umgebung zu verteilen und die Teilaufgaben effizient unter sich aufzuteilen. Das Ergebnis einer solchen starken Koordination war die Reduzierung der Explorationsdauer sowie eine generelle Stabilität des Systems.

Ein digitalisiertes dreidimensionales Modell einer komplexen Innenumgebung wurde mit Hilfe von zwei mobilen Robotern - ausgestattet mit verschiedenen Sensortypen - erstellt. Die Versuche wurden im ersten Obergeschoss des Rechenzentrums der Universität Karlsruhe (TH) durchgeführt. Drei unterschiedliche 3D-Modelle wurden akquiriert: Punktwolken, eine triangulierte Repräsentation und ein trianguliertes Modell mit abgeschnittener Decke. Bei diesem Experiment auf effiziente Weise eine geschlossene dreidimensionale Karte der komplexen Versuchsumgebung erstellt werden.

Inhalt der Arbeit

Im Gegensatz zu den meisten anderen im Bereich der Kartierung eingesetzten Multirobotersystemen verfolgt die vorliegende Arbeit den Ansatz einer expliziten Kooperation und Kommunikation zwischen den beteiligten Robotersystemen. Es zeigte sich, dass eine nichtkooperative Lösung z.B. durch Erforschung unterschiedlicher, klar getrennter Bereiche zwar die Komplexität des Systems gering hält, aber in den meisten Fällen nicht die effizienteste Aufgabenverteilung zwischen den Robotern darstellt. Eine Kooperation zwischen den Robotern zur effizienten Aufgabenverteilung und der dazu benötigte Austausch von Informationen verbessern die Effizienz des Systems deutlich. Es war also im Rahmen dieser Arbeit nicht nur eine Rahmenarchitektur zur Koordination eines Multirobotersystems zu entwickeln, sondern auch eine Erforschungsstrategie, die es erlaubt, zur Lösung einer Explorationsaufgabe jedem Roboter unterschiedliche Teilaufgaben zuzuweisen.

Kapitel 2 untersucht den Stand der Technik im Bereich der Roboterexploration. Schwerpunkte der Darstellung der einzelnen Arbeiten sind jeweils sind die verwendeten Sensortypen, die Repräsentation der Karte, die Erforschungsstrategie, die Genauigkeit des verwendeten Algorithmus ist und die Erzeugung des zugehörigen 2D- oder 3D-Modells. Zusätzlich werden

verwandte Arbeiten in den Bereichen „*Kooperationsaufgaben*“ sowie „*Kommunikationsprotokolle*“ vorgestellt und diskutiert.

Kapitel 3 gibt einen Überblick über das System zur 3D-Exploration mit einem Mutirobotersystem. Dargestellt werden die Einschränkungen bei der Verwendung eines Multirobotersystems und die Möglichkeiten, wie solche Einschränkungen mit einem Master-Slave-Ansatz vermieden werden können. Außerdem werden die Struktur der mobilen Robotersysteme und die Steuerungsarchitektur genauer betrachtet. Zuletzt stellt das Kapitel die im Rahmen dieser Arbeit erstellte Entwicklungsumgebung vor.

Kapitel 4 präsentiert die eigens entwickelte Explorationsstrategie für Multirobotersysteme. Drei Erwartungswerte für eine effiziente Exploration der Umgebung werden vorgestellt: die Navigationskosten, die Sichtbarkeit relevanter Merkmale und der Koordinationsfaktor. Diese Werte werden benutzt, um eine globale Bewertung aller potentiellen Zielregionen zu ermitteln. Versuchsergebnisse für die drei Fälle Einzelroboter, zwei Master sowie mehrere Slaves werden vorgestellt.

Kapitel 5 behandelt die Koorperationsstrategien kooperierender Roboter. Zunächst wird ein Überblick über die kooperative Robotik und Kommunikation gegeben und eine Klassifikation von Multirobotersystemen basierend auf ihrer Koordinationsstrategie vorgestellt. Aktive und passive Kommunikation werden vorgestellt, um mögliche Kooperationsstrategien zwischen Robotern zu zeigen. Zuletzt werden unterschiedliche Kooperationsstrategien anhand von Simulationsergebnissen mit einer Master-Slave-Struktur erläutert und validiert.

Kapitel 6 befasst sich mit dem Algorithmus zur Merkmalsextraktion bei natürlichen Landmarken unter Verwendung von zwei unterschiedlichen Roboterstrukturen. Das Farbraummodell und ein Segmentierungsansatz aus der Literatur werden definiert. Die Segmentierungsmethoden Region-Growing und Mean-Shift werden mit ihren Vor- und Nachteilen präsentiert. Das Konzept des Merkmalsextraktionsalgorithmus wird durch Versuchsergebnisse aus Experimenten mit zwei mobilen Robotern und unterschiedlichen Sensorsysteme dargestellt und validiert.

Kapitel 7 stellt eine Leistungsbewertung eines Master-Slave-Systems vor, die im Rahmen dieser Arbeit entwickelt wurde. Sie betrachtet zwei Hauptkriterien: die Explorationsdauer und die Koordinationseffizienz des Systems. Zunächst erfolgt die Evaluation in Bezug auf die Kooperation zwischen Master und Slaves, danach in Bezug auf den verwendeten Algorithmus zur Merkmalsextraktion. Schließlich wird der im Rahmen dieser Arbeit entwickelte Algorithmus durch Experimente in einer komplexen Umgebung validiert.

Chapter 1

Introduction

1.1 Motivation

Autonomous mobile robotic systems have gained a great interest in the last years because of their potential for taking over tasks that formerly humans had to perform. They could be used not only in industrial and household applications but also in dangerous situations that are too risky for human intervention. Examples of those situations include applications where toxic or nuclear wastes could threaten a person's live, rescue and search missions and extra-planetary exploration.

Inherent to many robotic applications is the need to explore the world in order to effectively reason about future plans and objectives. In order to operate and perform complex tasks in an unknown environment, robots must be able to collect information and understand their surroundings. In order to effectively explore an unknown environment, it is necessary for an exploration system to be reliable, robust and efficient.

Some of the important future applications of mobile robot systems are autonomous indoor and outdoor measurements of buildings, factories and objects in three dimensional view. A goal of such a measurement is to provide us with a detailed map of the environment with the interesting characteristics. This map can then be transferred into a model, which represents the measuring objects. By following an automated progress, an autonomous measurement robot could be useful in order to extract independently the map and model of the environment. The environmental model can then be directly used by the autonomous mobile robot for navigation. A further increase of the effectiveness in map production can be achieved, if the environment is mapped by several robots, or whole robot fleets. Such an approach can be advantageous, for example, by reducing the exploration duration. There are several reasons why two or more robots can be better than one:

- Distributed actions: Multiple robots can be in different places at the same time, and can localize themselves more efficiently if they exchange information about their position whenever they sense each other [Fox 99a].
- Inherent parallelism: Multiple robots can do many, perhaps different things at the same time, and thus have the potential to finish a single task faster than a single robot [Burgard 02].

- Simpler is better: Often each robot in a team of robots can be simpler than a more comprehensive single robot solution. Therefore, they can be expected to be more fault-tolerant than only one powerful and expensive robot.

The drawbacks in multi-robot approaches lie in the areas of coordination and elimination of interferences. Also, the overall system will be very complex.

A good exploration strategy is one that generates a completely accurate map in a reasonable amount of time. Robotic mapping has been a highly active research area in robotics for at least two decades. It addresses the problem of acquiring spatial models of physical environments through mobile robots. The mapping problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots. At present, there are robust methods for mapping environments that are static, structured, and of limited size. Mapping unstructured, dynamic or large scale environments remains largely an open research problem.

The map representation is such that it must be constructed from sensor data by the robot. The position of the robot must be known to build the map using sensor data from the robot. For the robot to be autonomous, it must perform both pose estimation and mapping. However, since pose estimation requires a map and mapping requires the pose, there is a "Chicken and Egg" problem. Which comes first? The answer to this question is that they have to be carried out at the same time. The process of building a map at the time as estimating the pose of the robot is called Simultaneous Location and Mapping (SLAM). SLAM is different from ordinary map acquisition since the uncertainty in the robot pose is accounted for when building the map.

Almost all existing algorithms for acquiring such maps operate in 2D, although there is a need to model indoor environments with mobile robots with volumetric 3D models, which would have two important advantages: Firstly, 3D maps facilitate the disambiguation of different places, since 3D models are much richer than 2D models and hence possess fewer ambiguities. Secondly, they are of particular interest if the goal of mapping goes beyond robot navigation. 3D maps interpret the true nature of indoor environments, which are often composed of walls, doors, windows etc. An understanding of such objects and their geometric properties will easily help the mobile robot in the navigation.

Using the extended possibilities of a fleet of mobile robot systems to effectively create a well-prepared map, the question raises which characteristics this map should contain. Geometrical information alone is often not sufficient to plan and accomplish robot-based actions. In order to be able to accomplish a successful classification of objects, colour and a prior structural knowledge about some features are very important. Only by these additional information can mobile platforms meaningfully classify objects on different abstraction steps and accomplish actions according to their mission.

To fulfil the goals provided by these challenging applications mentioned in last paragraphs, robots generally have to deal with unknown dynamical environments. For this reason, the complexity of the task could be too high to be achieved by a single robot. Teams of cooperative robots are necessary in many cases to avoid the expensive design of a single robot to solve the task. Moreover, the time is a critical factor to efficiently solve a task and has to be considered, especially in time-demanding applications like spatial exploration. Using several

cooperative robots instead of a single robot is one of the possible solutions to the temporal constraints and helps to increase the performance of the system. All these arguments were in favour of the use of cooperative multi-robot systems which explains the large amount of scientific work and experiments in this field.

Because of this increasing use of cooperation between robots, the question of the necessity of the communication has arisen. The communication problem became tightly related to the cooperative aspects because cooperation generally assumes an exchange of information between robots. The form and content of the communication depend on the task. There are typically two forms of communication, an implicit and explicit one. The implicit one consists of observing the action and pose of a second robot in order to decide about the own action or to correct its own pose estimate. The explicit communication consists of sharing sensors data and status information using explicit communication devices.

1.2 Problem Statement

Simultaneous Localization and Mapping (SLAM) is an area which has been studied by different researchers in the past few years. The various approaches have been implemented in order to overcome some algorithm problems. Firstly, SLAM algorithms in general suffer from the uncertainty in the position of the robots, features and objects in the environment. This uncertainty tends to lead to an inaccurate map of the environment. This is due to some factors. Amongst others the errors in the odometry sensor readings lead to a high increase in uncertainty. Some robots equipped with position sensors such as GPS (Global Positioning System) can reduce this uncertainty in the position reading, but they are usually not applied in indoor exploration. In contrast, robots equipped with range sensors such as laser scanners, cannot get correct absolute position of the robots and the features of the environment without any assistance of some artificial landmarks. The second problem is the computational and storage required by the algorithm, in order to run the algorithm online, and let the robots be truly autonomous. The third problem leads the SLAM algorithm to diverge from the optimal solution and to get inconsistent map, when extracting a feature using a weak algorithm of the data association.

These problems could be simplified by using multi mobile robots and 3D map facilities. This is done in order to facilitate the disambiguation of different places. This will have a number of consequences: reducing the uncertainty in the robot position by using passive exploration, reducing the computational and storage required by the SLAM algorithm through using a master-slave approach and dividing the environment hierarchically to generate sub-map regions, and then form a map for each sub-map region separately.

Most of the systems applied for mapping do not use explicit cooperation and communication to avoid the complexity of the cooperation aspects. The simple solution would be in these non-cooperative cases, to let the robots explore different areas. These areas are divided between them in relation to their mapping speeds to make them finish their tasks at the same time. Each robot has to solve its part of the task without knowing any information about the status of the other robots. Such a partitioning of the areas sounds to be an optimal solution, but this is only the case if the robots are not to be robustly cooperative. So this

way of splitting the task among the robots is not as efficient as it could be, and could be improved if the robots cooperate and help each other while doing their tasks. In order to improve the cooperation, all robots have to exchange information in order to achieve the tasks very quickly. Thus some form of communication protocol is needed to realize this cooperation.

Coordination is an essential characteristic of any task-achieving multi-robot system, whether it is accomplished through an explicit or implicit coordination mechanism. There is little formal work addressing how various coordination mechanisms are related, how appropriate they are for a given task, what capabilities they require of the robots, and what level of performance they can be expected to provide. So, there is a need to design a framework architecture which allows the development of multi-robot system that can manage the coordination problem efficiently.

Developing an exploration strategy using multi mobile robots is another problem to be solved. For this purpose, different tasks have to be assigned to each robot in order to implement the tasks as efficiently as possible.

Finally, some other problems have to be solved to achieve the goals stated above, for example, when a stereo camera is used to extract some landmarks for the navigation process. An approach needs to be designed, which consists of the following modules: data acquisition, pre-processing, segmentation and landmark extraction and characterisation. Such an approach should allow the use of multiple models to describe the measurement process for different parts of the environment and avoid the data association effect. Furthermore, by using a rotated laser scanner to represent the environment as a range image data, another difficulty exists in the consistent fusion of different range images with one another and in their integration into the environmental model. Moreover, the model must be hierarchically structured in order to be able to deal with large environments efficiently. In doing so, the memory requirement for the model, its real time capabilities and fast data fusion have to be taken into account.

1.3 Main Contribution of the Thesis

This thesis is concerned with the development of an efficient 3D map building algorithm for indoor environments using multi-mobile robots. For the various problems to be solved in this context, we employ methods from different research area such as mobile robotics, optical measurement techniques, computer vision, and cooperation and communication aspects. As a main goal, a 3D map of the indoor environment is to be built autonomously within a good exploration time by using multi-mobile robots. During the process, the exploration algorithm should choose the motion commands in order to achieve the tasks by the robots very quickly. A good collaboration among a team of robots is achieved in order to let them solve efficiently the exploration task. As a result, the overall time needed to complete the exploration mission is expected to be reduced. The coordination among them is taken into consideration so that they efficiently distribute themselves over the environment avoiding redundant work and reducing the risk of interference between the vehicles.

Thus the main contributions of the thesis are as follows:

- **Design of a framework for 3D mapping using multi-mobile robots**: We suggest an approach to distribute the robots over the environment, in order to avoid redundant work. A master-slave approach is used to coordinate among them. A hierarchy of one master and n slaves is implemented to achieve the exploration task efficiently. Theoretically, this hierarchy is extended to 2 masters with n slaves assigned to each of them. A theoretical study of this hierarchy is performed showing the advantage of having such a structure. Different sensor capabilities are used by the master and the slave according to the type of work they are supposed to carry out.

- **Development of a motion command algorithm**: We describe how to distribute the robots over the environment. This is achieved using the master-slave hierarchy. The main objective of this algorithm is to let a robot move to a new unexplored place, based on certain prospect values of this place. One of these prospect values is the natural feature quality available at that place, whereas the second prospect value is the navigation cost to it.

- **Cooperative strategies among the robots**: A detailed simulative analysis of different cooperative strategies among the robots for both master and slave is performed, in particular with regard to identifying the advantages of using multi-mobile robots and a master-slave hierarchy. To apply one of these cooperative strategies in the real world, some communication protocols are implemented in order to interchange the measurement data among the robots. Furthermore, we studied passive communication, in order to let a robot estimate the position and orientation of its team using different sensor capabilities.

- **Evaluation of the experiment results**: Meaningful experiments are carried out for most aspects of the work in a real environment by using 2 mobile robots with different sensor capabilities onboard. Some simulative experiments are also conducted to evaluate the benefit from different cooperative strategies.

Furthermore, to achieve these main contributions efficiently, some additional methods and tools are necessary, for example:

- **Implementation of a feature extraction algorithm**: Natural landmarks available in the indoor environment have to be identified based on some predefined knowledge about their features, as for example the texture behaviour or geometrical structure. Within this work, a segmentation process is developed and implemented by using region growing techniques. This type of segmentation allows the algorithm to be used online. Furthermore, the quality of the feature extraction algorithm is improved by applying some classification rules to identify the located feature. As a result, a point feature is obtained by using a corner detector algorithm. It is used by the SLAM algorithm to estimate and match the pose of the robot.

- **Fusion of different range image data**: As a result of the motion command algorithm, the slave is to be sent to different targets in the environment, in order to build up a 3D map of that place. The data map is in the form of range image data. Through the communication channel, the master receives a lot of sensor data in form of range

image data of different places from different slaves, and needs to fuse this data into a global map of the environment.

- **Implementation of a SLAM algorithm**: A reliable algorithm for simultaneous localization and map building in an unknown environment is achieved by using information from the measurements of arbitrary features and the robot odometry. Within this work, the extended Kalman filter is used to match the features in order to recover an accurate position estimate of the robot.

1.4 Structure of the Thesis

The thesis is organized in eight chapters. After this introductory chapter, chapter 2 reviews thoroughly the most relevant state of the art related with cooperative multiple robot systems and the field of robotic exploration.

Chapter 3 proposes a system overview in order to perform 3D exploration using multi mobile robots. It provides the reader with the basic concepts of the master-slave structure. Then the structure of the mobile robots and its software architecture used within the work are presented. The framework architecture is then presented, in which our work is situated.

Chapter 4 proposes the exploration strategy algorithm, which is presented in case of one robot or of two robots used as master and slave combination. Several prospect values are presented in order to let the robots explore the environment efficiently. At the end, the experimental results are presented taking into consideration three cases such as one robot, two masters or several slaves.

Chapter 5 describes the cooperative strategies with teams of cooperative robots. It is started with a definition of a cooperative robotics and communication, showing as well the taxonomy of multiple robot systems focused on coordination. Active and passive communication are presented, showing how can the cooperation done among the robots. At the end, the cooperative strategies are illustrated and validated through the presentation of results obtained with a simulation experiments using a master-slave structure.

Chapter 6 focuses on the algorithms used to extract a natural landmark using two different robot structures. It first defines the colour space model and the segmentation approaches available in the literature. Then the chapter presents the region-growing and mean shift segmentation methods. The chapter outlines as well the advantages and disadvantages of each approach. Further, the concept of feature extraction algorithm is illustrated and validated through the presentation of results obtained within experiments using two mobile robots equipped with different sensors.

Chapter 7 proposes the performance evaluation of the developed master-slave system showing its merit, limitations and prospects.

Chapter 8 concludes the thesis. It makes a summary of the work, underlines the main conclusions obtained in the course of the research herein reported, and discusses the advantages and limitations of the presented contributions. The chapter ends by pointing out perspectives on future research.

Chapter 2

State of the Art

2.1 Introduction

Robot navigation is the task of an autonomous robot to move safely from one location to another. The general problem of navigation [Borenstein 96], [Siegwart 04], [Thrun 06] can be formulated in three questions,

- *Where am I?* The robot has to know where it is in order to make useful decisions. Finding out the whereabouts of the robot is called *robotic localization*.
- *What does the world look like?* In order to fulfil some tasks, the robot has to know how its surroundings look like. It has to interpret its sensor data into a suitable representation to help it to achieve these tasks. This problem is known as *mapping*.
- *How do I get there?* Once the robot knows where it is and where it has to go to, it has to decide on how to get there. Finding a way to get to the goal is known as *motion control*.

Makarenko et al. [Makarenko 04] definition of the field of robotic exploration is based on those three questions above. Figure 2.1 illustrates this definition with three partially overlapping regions, namely localization, mapping and motion control. In order to make robots navigate autonomously, existing solutions provide a varying degree of integration between these three regions. The integration of motion control and mapping is known as active exploration, which encompasses research on strategies [Schultz 99], [Simmons 00], [Yamauchi 97], [Moorehead 93], [Lee 03], whereas the integration of localization and motion control is the field of the active navigation [Fox 98], [Roy 99], [Thrun 98a], [Kaelbling 96]. The intersection of localization and mapping constitutes the area of SLAM algorithms [Dissanayake 01], [Feder 99], [Leonard 99], [Thrun 98c], [Smith 87a]. Full integration of all three tasks is the goal of this work, and will be referred to as Robust Exploration.

In this chapter, current work within the different sub-areas of Robust Exploration is presented and discussed. The focus in this overview is on the type of the used sensors, on how the map is represented, on how the exploration strategy is done, on how accurate the used

algorithm is, and on how a 2D or 3D environment model is accomplished. In addition to the state of the art in the area of Robust Exploration, the last section of this chapter presents and discusses the related work on cooperative tasks and communication protocols.

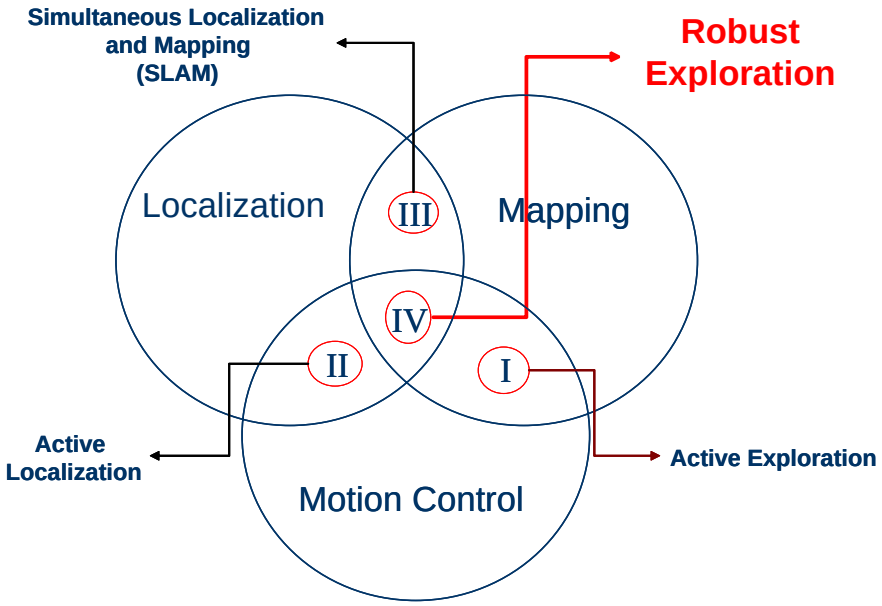


Figure 2.1: The field of robotic integrating different areas of research

2.2 Active Exploration

The process of map building starts with perceiving information about the environment through the sensors to build a map. To build a complete map of the environment, it is necessary to plan the motion of the robot in order to guide it through the environment. Such approaches are known as active exploration.

Our work is concerned with the domain of indoor exploration, which has been studied by several researchers [Kuipers 91],[Mataric 92]. Most of them used the structure of the indoor worlds, such as parallel and perpendicular walls, as help in solving the exploration problem.

Yamauchi in ([Yamauchi 97] and [Yamauchi 98]) presented a frontier-based robot explorer designed to explore complex environments typically found in office buildings. The premise of this method is that "*To gain the most information about the world, move to the boundary between open space and uncharted territory* " [Yamauchi 97]. Within this work, the robot plans the shortest path to the nearest frontier, without taking into consideration the amount of information it will receive from this target. It may in fact be better to guide the robot to a slightly further frontier if it is larger than a closer one.

Simmons et al. [Simmons 00] used multiple robots to explore using a similar frontier strategy. However, each frontier is evaluated based on the expected highest utilities. In this work, the frontier evaluation values were the information gain and the driving cost. So, the robot plans the path to the best evaluation frontier rather than simply to the closest one. The multiple

robots coordinate their efforts to reduce overlap while exploring. Comparing to our work, different evaluation values were used such as the localizability prospect and coordination factor. Natural landmarks are used to measure the localizability factor, while the coordinator factor prospect is used to distribute the robots efficiently over the environment.

Rocha et al. [Rocha 05] conducted a study to control the motion of a team of robots, each of which is equipped with a stereo camera (see figure 2.2). Each robot is able to build a 3D map out of measurements from its own stereo-vision sensor and is committed to cooperate with other robots through information sharing. Within this work, a 3D discrete grid [Moravec 89] which divides the work space in equally sized vowels, is used to represent the environment. A gradient-based strategy is applied which drives a robot to regions with higher entropy. Based on entropy information utilities, the robots cooperate with each other on building a 3D map. In their work, the authors did not introduce any coordination technique to avoid the robots to overlap while exploring. They assumed that the localization of the robot at every point is known, which is introduced by some external efforts.

(a) (b)

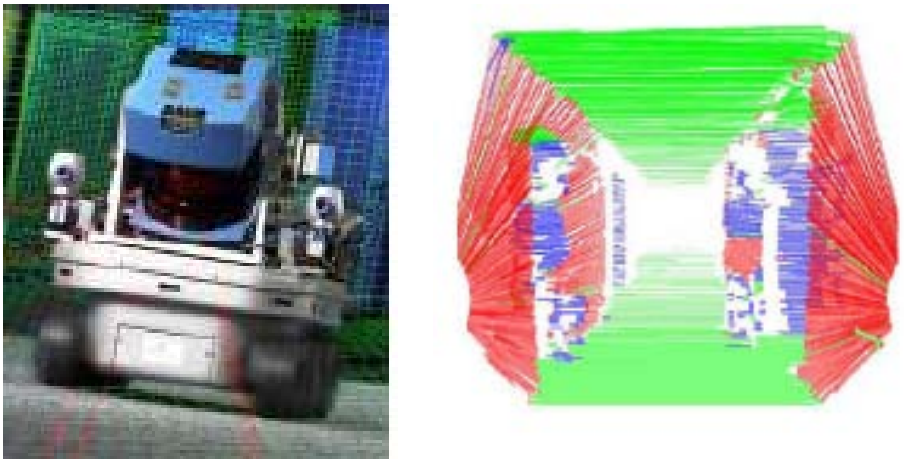
Figure 2.2: Two cooperative mobile robots building a volumetric map used by Rocha (a) two robots equipped with stereo vision, (b) example of a volumetric map

The research presented above uses the technique of evidence grids to motivate exploration. On the contrary, Newman et al. [Newman 03] presented an exploration algorithm which uses a feature-based world representation to deduce robot trajectories that are likely to expand and refine the map of the robot's environment. Within this work, a truly autonomous exploration and mapping is demonstrated running in real time in an indoor environment. The authors assumed that the robot is equipped with a SLAM algorithm that is capable of consistent mapping and localization.

Beyond grid-based and feature-based approaches, there is also the possibility to represent a map based on topological approaches. Berns et al. [Berns 05] demonstrated their work by using such a representation in order to produce an abstract topology-based map for the autonomous exploration. Within this work, the use of this representation proved meaningful, since it simplified the localization and navigation for mobile robots. Other research presented by [Fabrizi 02] and [Thrun 96] tried to generate a topological map from a grid map to be used in the navigation process. Finally, there is also the possibility of integrating of both map types in order to use the advantage of both procedure, as represented by [Thrun 98b].

Surmann et al. [Surmann 03] proposed an exploration method which uses a so-called next-best-view algorithm to calculate a position which promises a maximum of new information

about the environment. In this work, an automatic system for digitization of 3D indoor environment without any human intervention is designed and presented. An autonomous mobile robot together with a 3D laser range finder sensor are used in order to get a 3D digitalized model of a large indoor environment as shown in figure 2.3.



(a) (b)

Figure 2.3: Autonomous modelling of indoor environments done by Surmann (a) mobile robot equipped with 3D laser scanner, (b) example of 3D model

In summary, there are three main methods discussed in the area of the active exploration to represent the map of the environment, namely topological, grid-based and feature-based approaches. Each of them has some advantages and disadvantages, and sometimes a combination of these approaches is used to combine the benefit of advantages of the different methods. The topological maps have the advantage of using a small amount of memory and of recording only those parts of the environment that are of interest. Therefore, topological maps scale well to large environments. Grid-based maps are computationally expensive and require comparably more memory in large-scale environments. Despite that, they are sometimes preferred, since they can hold more detailed information than a topological map. They can also be more useful for performing tasks such as path planning and obstacle avoidance. Feature-based maps, on the other hand provide an elegant way such as extended kalman filter based approaches to manage uncertainty of localization. However, these approaches suffer from the data association problem. Almost all existing algorithms for acquiring such maps operate in 2D by using some inexpensive sensors like ultrasound or sonar sensors. Few research works exist [Nevado 04][Thrun 00] which build such maps in a volumetric 3D Model. These types of maps have two important advantages: First, 3D maps facilitate the disambiguation of different places, since 3D models are much richer than 2D models and hence possess fewer ambiguities. Second, they are of particular interest if the goal of mapping goes beyond robot navigation, e.g. for surveying, architecture, rescue and search missions.

2.3 Active Localization

The problem of robot localization consists of answering the question *where am I?* from a robot's point of view. This means the robot has to find out its location relative to the

environment. In most algorithms, this includes some X and Y coordinate along with an orientation angle that represents the robot position. Most algorithms of localization are supposed to have a correct world map. Different algorithms are implemented using different approaches based on the type of the given problem instances. For example, in a position tracking problem the robot knows its initial location. The goal of the localization here is to keep track of the position while the robot is navigating through the environment. Techniques that solve this problem are called *tracking* or *local techniques* [Fox 99b], whereas methods that solve the global position problem are called *global techniques*. Within these methods, the robot does not know its initial position, and it has to localize itself from scratch [Se 01].

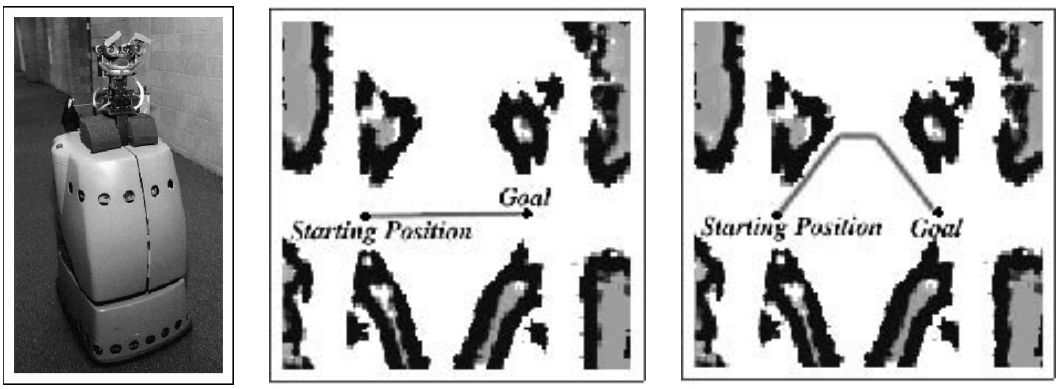
A widely used but not very dependable way to localize a robot is odometry. Odometry involves recording the turns of the wheel and from this estimating the distance travelled. This approach is often used because it is very inexpensive, simple and easy to implement. It has a fair dependability in some indoor environments, but in outdoor situations wheel slippage renders it close to useless. Also, the initial position of the robot must be known. This limits the abilities of the robot to be autonomous. For these reasons, in most of the recent research odometry is not used for localization.

The general localization problem can be described as a Bayesian estimation [Bruyninkx 02]. Fox et al. [Fox 98] derived a Bayesian framework that estimates the probability density over the space of all locations, which is called Markov Localization Framework [Fox 98]. In this work, a topological map is used as environment representation. A Markov model can be thought of as a set of states and the probability that the robot is in a particular state [Burgard 96].

Roy et al. [Roy 99] used natural landmarks for path planning and robot localization. This produces paths which remain close to walls. Figure 2.4 shows a particular problem of motion planning demonstrated by Roy. A robot performs a coastal path planner instead of shortest path planner in order to guarantee that a mobile robot can reach its goal with maximum reliability. As they do not try to explore new regions, two criteria are considered for path planning shortest distance and localization ability. In which, finding new terrain and finding good localization places are considered the two possible goals of doing the exploration. Many other researchers have considered these goals in their work as well e.g. [Thrun 98d] and [Thrun 00], however more attention is usually given to the problem of localization than to deciding which paths to take in the exploration.

In some approaches, easily recognizable objects are placed in the environment which are called artificial landmarks [Singhal 97]. This approach has been used for localization with success in limited environments. However, it is far from optimal because the goal of an autonomous robot is to be able to learn about its environment on its own. One more disadvantage is that the further a robot is away from a landmark, the less accurate the position estimation becomes.

A number of methods have been developed using different representations for the localization process. Topological graphs, grids, particle filters and kalman filters are some of the approaches which were used. Kaelbling et al. [Kaelbling 96] used a topological representation in order to localize the robot with a limited set of robot actions. By using this representation, the global localization problem is solved. However, the resolution of the representation in nodes and arcs is coarse and thus the accuracy of the localization estimates can be low.



(a) (b) (c)

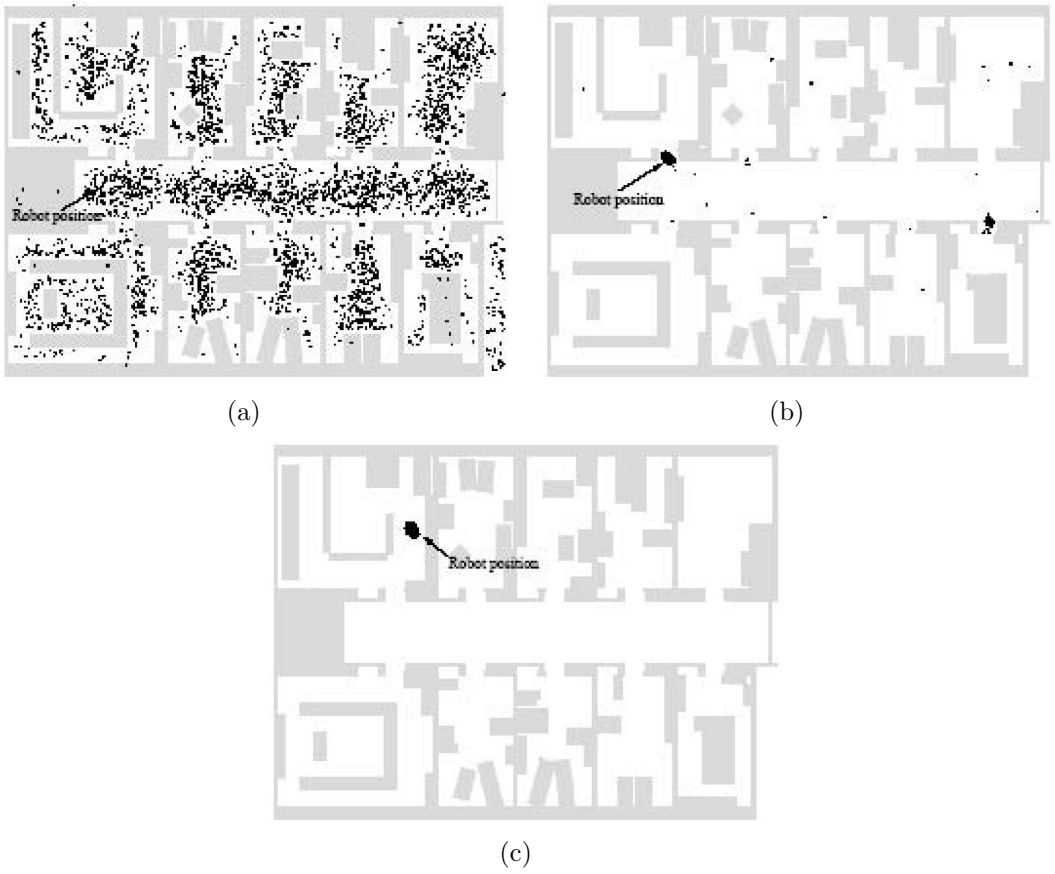
Figure 2.4: Ability of the localization in different areas of the environment (Roy at el.) (a) mobile robot use by Roy, (b) Shorest path planner, (c) Coastal path planner

Burgard et al. [Burgard 96] showed how a robot can localize itself using probability grids given the world map. A probability grid is simply a two dimensional array-like structure containing data relating to the outside world. This has the advantage that the localization works also in environments where no identifiable features exist. However, the complexity of the sensor models increases, because a probability has to be maintained for every raw sensor measurement.

Yet another way of solving the localization problem is by representing the position estimate by a set of m weighted samples distributed according to the position estimate [Fox 01]. This recent technique is known as particle filter method. In [Fox 01] such a filter is used and evaluated. The localization results achieved with this technique have shown to be more efficient and accurate than previously mentioned methods (see figure 2.5).

Kalman filter techniques are another method to represent the location space continuously as a parameterized probability density. They are developed and used by different researchers [Grewal 93], [Kuipers 91], [Leonard 99], [Moutarlier 89] and [Smith 87a]. Such a filter can be used efficiently to solve the position tracking problem, since the computations involved are greatly reduced. However, being unimodal distribution with one peak, like the Gaussians they do not allow more than one hypothesis about the location.

To conclude what is stated in the literature concerning active localization, two sub-problems have to be taken into account, position tracking and global positioning. To perform localization, a robot needs to have access to prior knowledge and navigational information. The prior knowledge comes in the form of maps, whereas the navigational information consists of relative and absolute measurements. Therefore, most research approaches the localization problem from a probabilistic point of view. Within such research, different localization methods were developed, either by representing the location space continuously as a parameterized probability density like a Gaussian, as in a kalman filter, or by discretizing the location space and representing it as a topological or grid map. Grid-based approaches are more accurate than topological methods, but they come at higher computational costs. Both kalman filters and particle filters give a very accurate performance, and have been successfully applied in SLAM and Fast SLAM algorithms. The specific advantages and disadvantages of each approach need to be considered depending on the desired application.



(a) (b)

(c)

Figure 2.5: Global localization of a mobile robot using MCL (10,000 samples) (Fox et al.)

2.4 Simultaneous Localization and Mapping (SLAM)

The research discussed in the previous two sections treated map making and localization as being two separate and unrelated problems. Within this research, the problem of localization given a map of the environment or estimating the map knowing the robot position has been addressed and solved using a number of different approaches. A related problem is when both, the map and the robot position are not known. In this case, the mobile robot starts in an unknown location in an unknown environment and proceeds to incrementally build up a navigation map of the environment while simultaneously using this map to update the location. A number of approaches have proposed to address the SLAM problem [Durrant-Whyte 00], [Guivant 01], [Leonard 91] and [Thrun 04]. The most popular of these approaches adopts the estimation-theoretic or kalman filter based approach. In this approach, a kalman filter is applied to implement the problem of simultaneously localizing the position of a robot and building a map of the environment. It directly provides two major points. The first is a recursive solution to the navigation problem. The second is a means of computing consistent estimates for the uncertainty in robot and map landmark locations. This is done on the basis of the statistical models for robot motion and relative landmark observation.

Dissanayake et al. [Dissanayake 01] have shown that it is possible for an autonomous robot starting in an unknown environment to incrementally build a perfect map of the world and to compute simultaneously a bounded estimate of the robot location by using a kalman filter-

based approach and by collecting relative observations only. Based on the theoretical results, they proved in their paper[Dissanayake 01] that uncertainty in the relative map estimates reduces monotonically, that these uncertainties converge to zero, and that the uncertainty in robot and absolute map locations achieve a lower bound.

Newman et al. [Newman 01] demonstrated a solution to the SLAM problem referred to as Absolute Mean Filter(AMF). This algorithm was first published by Smith [Smith 87a]. The AMF filter uses a Kalman filter to estimate a state vector containing both the robot and the landmark states in a global coordinate frame. Using the AMF filter, the three keys of convergence properties stated by Dissanayake [Dissanayake 01] are tested and it is proved that the SLAM problem can be solved. An important finding was the fact that the AMF filter suffered a scaling problem with a computational effort and storage requirements of order N^2, where N is the number of the estimate landmarks.

Csorba et al. [Csorba 97] introduced a suboptimal algorithm, called Relative Filter (REL), which avoids many of the computational and practical problems of the AMF filter. The idea of this filter is based on the relative distance between the features and their neighbours. This allows the REL filter to construct a consistent map without the need to maintain a large covariance matrix for all possible pairs of features as the AMF filter must. As a result, the storage requirement of the relative filter grows linearly with the number of features. Since no correlations have to be considered, the storage also grows linearly and the computational burden does not increase with the number of features. The main disadvantage of the relative filter is the absence of absolute position estimates. In particular, the REL filter can only localize the pose of the robot relative to the features.

Newman et al. in [Newman 01] and [Newman 02] have developed a SLAM algorithm, in order to solve the SLAM problem in real time. This algorithm is called the Geometric Projection Filter (GPF). Rather than to estimate the location of landmarks in global coordinates, it estimates the relationships between individual landmarks. Unlike the REL algorithm, the GPF algorithm is guaranteed to build a consistence relative map.

Williams et al. in [Williams 02a] and [Williams 02b] have developed a new approach to the mapping of terrain features that provides improved computational efficiency in the SLAM algorithm. This algorithm is called the Constrained Local Sub Map Filter (CLSF). Rather than incorporating every observation directly into the global map of the environment, the CLSF filter relies on creating an independent, local sub map of the features in the immediate vicinity of the robot. This local sub map is then periodically fused into the global map of the environment. Williams in [Williams 02a] and [Williams 02b] proved that the CLSF filter is able to reduce the computational complexity of maintaining the global map estimates. Within this approach, the update stage requires only $O(N_l^2)$ operations, whereas in the AMF filter, the update stage requires at best of $O(N^2)$ operations, where N_l is the number of the features in the local map and N is the number of all features extracted in the map. Thus as a result, $N_l << N$ which is a considerable saving for each individual observation.

One of the recent SLAM approach is presented by Montemerlo et al. [Montemerlo 02], which is known as FastSLAM. This algorithm does not require explicit loop-closing heuristics. FastSLAM follows a proposal by Murphy [Murphy 99] using a particle filter to sample robot poses and track the position of a fixed number of predetermined landmarks using a kalman filter. This method mitigates some of the challenges in mapping at the expense

of some challenges in landmark selection and identification. The latter can involve a fairly complicated data association problem, although recent progress has been made in addressing this [Montemerlo 03].

The previous research discussed so far uses 2D laser scanner sensors to solve the SLAM problem. Davison [Davison 02] however addressed the problem using a stereo camera, achieving good results by detecting and tracking suitable landmark features during goal-directed navigation. The use of line segments or planner patch features could extend the robustness of his approach by making reliable landmarks easy to find and to track in the environment.

Takezawa et al. [Takezawa 04] proposed a method for SLAM in an indoor environment using a stereo vision system. In their work, they presented a comparison between laser-based 2D SLAM and vision-based 3D SLAM. They used a stereo camera to achieve vision-based 3D SLAM. Specially designed artificial landmarks distributed in the environment are observed and extracted from a camera image. The disparity map obtained from the stereo vision is used to obtain the ranges to these landmarks. They showed that the artificial features used in their work improved the performance of the SLAM algorithm. Their work gives us the opportunity to use natural features in the SLAM algorithm. This is clearly an advantage in exploring unknown environments autonomously.

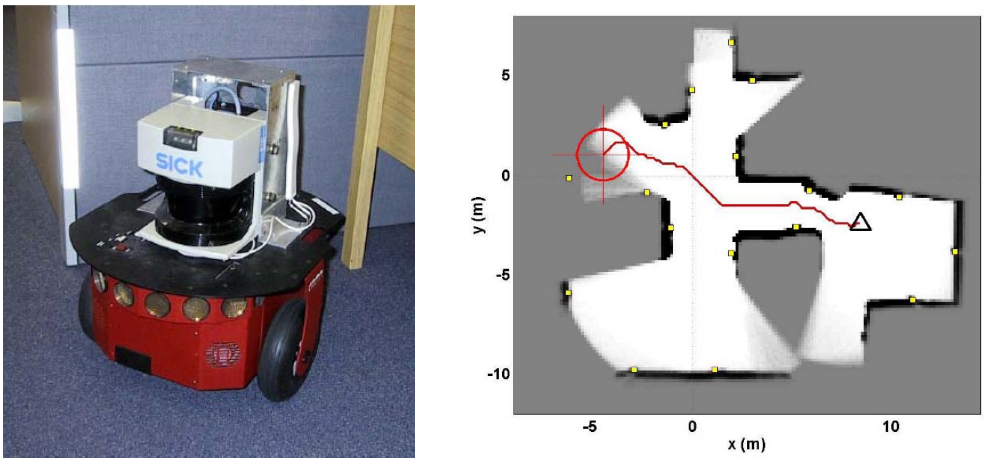
All approaches presented are aiming for an efficient solution to the SLAM problem. All of them try to reduce the computational load and storage required by the SLAM algorithm, in order to apply the algorithm in real-time applications. All of them used the feature landmarks as a Cartesian point (x, y), and only one robot to implement their solution. A CLSF algorithm could be modified to be implemented with multi robots, which can be helpful in order to achieve an efficient solution to SLAM problem by dividing the environment into sub-regions. Unlike exploring the entire environment at once, the sub-map region exploration procedure can reduce the computational complexity. In the same time, an extraction of 3D features can eliminate the problem that arise from data association algorithms, which can give an efficient performance of the SLAM algorithm.

2.5 Robust Exploration

As shown in figure 2.1, a robust exploration task is defined as addressing mapping, localization and motion control simultaneously. To get a robust exploration, the robot is enabled by the exploration strategy to move autonomously through its environment, while at the same time building a map by processing the relevant sensor data. Whenever the robot is moving, it considers actions to improve its localization, to acquire more information about unknown places, and to improve the exploration time by avoiding redundant work. In the end, the robot is assumed to have built an accurate model of the whole environment as well as to have determined its own pose relative to this model.

Several research groups focus on how to build such a strategy using some utilities function with different assumptions in order to perform a robust exploration. This is done using single robot systems as well as teams of robots. Makarenko et al. [Makarenko 04] pursued an exploration strategy for map building and localization using a single robot shown in figure 2.6(a). The robust exploration proposed in their work refers to a tight coupling

between the tasks of localization, mapping and motion control. Their strategy makes use of a utility function that evaluates the next robot sensing location. This utility function takes into account three elements: the possible information gain, the distance to sensing location (cost) and the utility of localizability based on a covariance matrix. The grid-based map is used to identify the region of interest based on the frontier region [Yamauchi 97]. The information gain utility is obtained from this map. The utility of the navigation is chosen in order to make the robot drives more appealing, whereas the utility of localizability is based on some artificial landmark distributed over the environment. Within this work, the SLAM algorithm is applied to generate and maintain a map of environmental features and at the same time to estimate the robot's pose using relative observations of the map features.



(a) (b)

Figure 2.6: Simulated path of an exploration mission to the best destination of highest utility function (Makarenko et al.) (a) Mobile robot used by Makarenko, (b) The map of the environment

Tovary et al. [Tovary 06] presented a motion planning approach for building a map of the environment. The main goal of their work was to determine the best robot target at every single iteration of the algorithm. A utility function is designed to combine the geometric information with an intensive usage of the results obtained from the perceptual algorithms. Based on this utility, the robot plans motions in such a way that its certainty localization is minimized. At the same time, the motion strategy takes into account that the robot must discover unexplored environment regions while minimizing energy consumption. The general concept behind this research is similar to the work done by Makarenko et al. [Makarenko 04]. However, there are several differences: First, Tovary et al. did not use a grid-based map. Second, the utility function is much better at balancing opposite factors. And third, this approach considered multiple robots as well. However, the SLAM algorithm is not applied in this approach.

Stachniss [Stachniss 06] presented a decision-theoretic approach to guide a robot during an exploration. This technique uses the coverage posterior in the map to reason about the uncertainty of the robot about each location in the environment. his goal was to choose the viewpoint that minimized the overall uncertainty in the map model. Different exploration strategies are analysed, and it is indicated that a technique combining the maximum uncertainty reduction and the distance to be travelled yields the best trade-off between the

number of necessary measurements and the length of the resulting paths. Within this work, a coordination among several robots is achieved in a way to reduce the redundant of the tasks.

Comparing our proposed approach with those researches presented above, it is found that all previous researches used 2D maps to model the environment. The work proposed by Makarenko does not consider the multiple robots case. Further, they used an artificial landmark to measure the localizability function, which reduces the degree of autonomy of the exploration task.

The exploration strategy presented by Tovary did not take the SLAM problem into account. Further, they used multiple robots in their work, however they did not present any mechanism to coordinate among the robots.

The work presented by Stachniss used a multiple robot in order to perform the exploration. Within this work, he used the particle filter techniques to achieve the task. Further, he did some experiments with robots that have a limited communication range, and showed that as the communication range increases, the benefit of the coordination approach improved. However, he did not perform either the localizability function nor the coordination function into his evaluation function.

Within our proposed work, we intend to use natural landmarks for the purpose of calculating the localizability function. Further, we use one more function in the evaluation which is responsible for distributing the robots efficiently over the environment. All the research presented above used the information gain function as one of the evaluation functions to control the motion of the robots. Within our proposed work however, it is intended that this function is not used in order to be able to run the strategy online. The feature extraction algorithm and the information gain algorithm have a high cost in time and cannot be applied together in order to let the algorithm run online.

2.6 Cooperative and Communication Aspects

The use of multi-robots provides a lot of benefits over single robot systems [Cao 97]. First, cooperating robots have the potential to achieve a task faster than a single one [Guzzoni 97]. By using several robots, accuracy can be explicitly introduced based on redundancy so that such a team can be expected to be more fault-tolerant than a single one. One more benefit of robot teams arises from merging overlapping sensor information, which can help to compensate for sensor uncertainty. As a result, the map can be expected to be more accurate. Another possible benefit of using multiple robots is presented by Rekleitis [Rekleitis 00], where a robot with passive communication is used to improve the position estimate of the team of the mobile robots. Rekleitis [Rekleitis 01] and Fox [Fox 99a] presented how multi-robots can localize themselves more efficiently, especially when they have different sensor capabilities. There are two related aspects of the work which are presented here: first, the work on how to use cooperation and communication to improve the system performance, and second the work on cooperative tasks done in the field of exploring unknown environment.

A number of researches exist which aim to show the utility of using cooperation and communication in multi-robot systems to increase their performance. This research has mostly taken

place in the 1990s. In 1993, Tan [Tan 97] made a comparison between independent versus cooperative agents, and proved that cooperating agents can use communication to improve team performance by sharing sensor information, past experiences and learned knowledge. In 1995, Mataric et al. [Mataric 95] presented an approach that utilizes cooperation at three levels: sensing, action, and control, and takes advantage of a simple communication protocol to compensate for the robot's noisy and uncertain sensing. They have shown that the cooperative box-pushing method using a simple communication protocol is more effective than the strategies not employing cooperation and communication. Other research addresses the question of cooperation in distributed multi-robot teams like the one developed by Parker [Parker 98]. He developed the Alliance architecture for a fault-tolerant cooperation among heterogeneous robots. The Alliance architecture allows the team to complete its mission under some predefined conditions, taking into awareness the quality of the team's performance in terms of the time and the energy required to complete the mission.

Recently, the method of using communication to support the cooperation aspects has been widely considered. Chang et al. [Chang 03] have affirmed the beneficial rule of communication towards an effective teamwork and proposed a cooperative method to solve two types of foraging tasks, namely consuming and surrounding tasks. The robots cooperate by sharing a common world model and communicate their sensor data explicitly to each other. The experimental results presented in this paper not only have shown that reliable communication contributes to better teamwork, but that in many cases the availability of limited communication packets is sufficient to considerably enhance the cooperative strategy of the team.

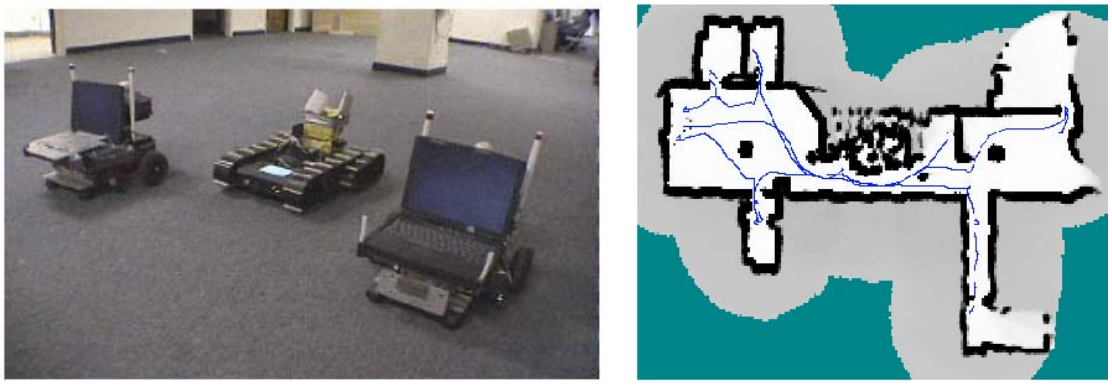
In [Wawerla 02], Wawerla et al. focused on the multi-robot coordination problem in cooperative construction. They presented experimental results with multiple, simulated robots, cooperating to do some task in the same environment, and compared the effects of communication and of two different environment sizes. Within this work, they showed that a minimal communication between the robots improves the system performance significantly in a construction task.

All these works have the same goal as one of the declared objectives for this present work. This objective is to improve the performance of a multi-robot system through cooperation and communication, despite that they are concerned with different tasks.

A considerable amount of research has been done in the area of exploring unknown environments using different mobile robots. For example, Singh et al. [Singh 93] presented a decentralized online approach for heterogeneous mobile robots in order to make an environment map. The used robots have different characteristics. They differ in their size, and in their capabilities in terms of speed to navigate through the region as well as in their sensor ranges to acquire information about the region. When a robot discovers an opening to an unexplored area that it cannot reach because of its size, it informs another robot which can carry out the exploration task in this area.

Yamauchi [Yamauchi 98] used a grid-based approach and frontier-based exploration to direct robots to the areas that are likely to provide the most new information about the world. The applied coordination strategy used allows multiple robots to share information and explore cooperatively without any human control. However, his approach has a few limitations. Since the robots navigate independently, they may waste time by navigating to the same

frontier, and may even physically interfere with each other. These limitation were avoided by Simmons [Simmons 00]. He used explicit coordination in order to keep the robots well separated and can thus significantly reduce the time needed to achieve the exploration (see figure 2.7).



(a) (b)

Figure 2.7: Coordinated exploration by a team of three robots in a real world experiment (Simmons et al.) (a) Mobile robots used, (b) The explored map of the environment

Rekleitis et al. [Rekleitis 00] focused on the problem of reducing the odometry error during the exploration. They separate the environment into stripes that are explored successively by the robot team. Whenever one robot moves, the other robots stay stationary and observe the moving robot.

In [Burgard 02] and [Burgard 05] the authors considered the problem of reducing the time processing of exploration by choosing appropriate target points for the individual robots so that they simultaneously explore different regions of the environment. They presented an approach for the coordination of multiple robots which takes into account the cost of reaching a target point and its utility.

Grabowski et al. [Grabowski 00] considered teams of miniature robots that overcome the limitations imposed by their small scale by exchanging mapping and sensor information. In this architecture, a team leader integrates the information gathered by the other robots. Furthermore, it directs the other robots to move around obstacles or to direct them to unknown areas.

In [Rocha 05], the authors extended their strategy to a team of multiple robots, which are committed to cooperate by sharing newly acquired sensory information. The robots interchange the information utility based on an entropy mechanism, without overwhelming communication resources with redundant or unnecessary information. This is done through sharing useful measurements. Each robot prepares a packet describing how the environment looks like that will be useful to the other robots. When a destination robot receives this packet of communicated measurement, it updates its local map as if measurements would have been gathered by its own sensor. So, the communication channel will always be limited in use. Within this work, they presented a performance comparison between using one robot and multiple robots in doing the mapping. They showed that the time of mapping is improved by using two robots rather than one robot. However, they did not use any form of explicit coordination within their work.

2.7 Conclusion

The chapter started with giving the definition of the area of exploration and requirements to let the robot navigate through the environment autonomously. The related work within the field of exploration is presented and discussed. The focus in this overview is on the type of the applied sensors, how the map is represented, how the exploration is done, and how accurate the applied algorithm is.

The first field of robotic exploration is the active exploration. In this section, a variety of research is presented and discussed, in particular three different methods of map representations: topological, grid-based and feature-based methods. Their advantages and disadvantages of each methods are illustrated.

The second field of robotic exploration is active navigation. Within this section, research is presented using different localization techniques, either by using a kalman filter approach or by using a particle filter approach.

The third field of robotic exploration is the field of the SLAM. Different approaches are presented in order to get an efficient solution to the SLAM problem. Most of them try to reduce the computational load and storage required by the SLAM algorithm in order to apply the algorithm in real-time application.

A partial combination of these three fields is referred to as robust exploration and is presented in section 2.5. Different utility functions were used in order to reduce the redundancy of the tasks.

Section 2.6 presents the related work on cooperative and communication aspects. Within this section, two different aspects are emphasized. First, the related work which uses cooperation and communication to improve the system performance is presented. This research shows that even the use of a simple communication can improve the system performance. The second aspect concentrates on related work concerning cooperation tasks in the field of exploring unknown environments. This work has been able to show that the time of mapping can be improved by using multiple robots.

Chapter 3

3D Exploration using Multi Mobile Robots – System Overview

3.1 Overview

Moving from single- to multi-robot systems increases the resources available to achieve a task but it also adds its own complexity as robots must be coordinated to function efficiently. Exploring an unknown environment is a task particularly well suited to multi-robot systems. Section 1.1 presented the advantages and disadvantages of using such systems over a single robot. In order to benefit from these advantages, and to avoid the disadvantages of multi-robot systems, several issues must be taken into account. For example, the robots must be distributed efficiently over the environment to avoid conflicting tasks and to reduce the interference from their sensors. Therefore, the coordination of multiple robots is the key to accomplish a task more efficiently, which is currently one of the main foci in robotics exploration research. Apart from coordination techniques, a framework system is necessary to autonomously perform an exploration task by multi-robot systems without any robot conflicts. This should be done cooperatively to enhance their task achievement performance taking into account the robot team capabilities.

As stated in section 2.6, the work done by [Yamauchi 98] and [Rocha 05] used some evaluation function to direct the robots to the area that is likely to provide the highest amount of new information about the world. However, these approaches suffer from a weak coordination, which leads to robot conflicts and sensor interferences in some situations. This limitation was improved by [Simmons 00] by using explicit coordination to keep the robots well separated. Explicit means that in such situations, the robot receives an external command to avoid such cases, which of course reduces the degree of autonomy. This type of coordination is classified by Farinelli [Farinelli 04] as strongly centralized coordination systems. Some exploration applications may require robots to work without any human intervention for a long time. Therefore, the system should be able to handle unpredictable situations and execute the demanded tasks while maintaining the cooperation level with its team-mates.

In [Grabowski 00], a multi-robot system is developed, in order to reduce the development cost of the overall application by using heterogeneous team members. Within this work, a

team leader is used equipped with very powerful systems. Several Millibots are used as its team equipped with very cheap sensors. They collaborate to map unknown environments controlled by the team leader who performs the high level planning. However, there are some limitations by using such Millibots. For example, with their small size comes the disadvantage of a limited mobility range, limited available energy, and possibly reduced sensing, communication and computation abilities due to size and power constraints.

This thesis proposes a solution to these limitation with a master-slave framework. The framework is composed of algorithm-based software modules. The concept is based on a hierarchical master-slave approach, where in some degree of autonomy of the slaves, the master directs its team in order to build up a 3D map. The overall system allows a set of autonomous robots not only to perform their task in a coherent and nonconflicting manner but also to cooperatively enhance their task achievement performance taking into account the strategy of exploration they are supposed to follow.

3.2 Basic Concepts of the Master-Slave Structure

To cooperate, the robots need a strategy to negotiate each robot's work. They need communication to exchange useful information, and they need coordination lest some potential conflicts may happen. At the end, they also need to fuse their local map into a global map. In the whole process of completing the task, the type of cooperation concept depends on which robot(s) are making the relevant decisions: *who and how to plan the work; how to communicate; who and how to solve the conflicts; who and how to build the global map.* One of the cooperation concepts is to assign a *commander* who decides all these points. This type of cooperation is called *master-slave cooperation.*

The master-slave cooperation concept refers to the following type of robot behaviour: one robot in the multi-robot system is to decide the task arrangements of all robots and coordinates all of them. All other robots in the multi-robot system obey the commands or the arrangements of this so-called "master". These other robots are called "slave robots". Figure 3.1 presents the simple relationship between the master and a slave. In this situation, the master sends its commands or arrangements to the slave, and the slave will act based on the master instructs, providing feedback to the master.

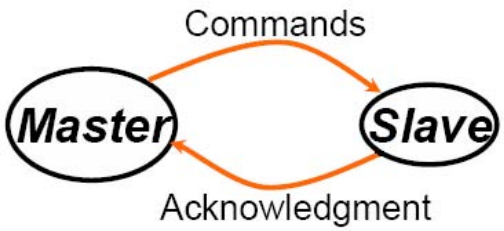


Figure 3.1: A simple master-slave relationship

Sometimes, simple master-slave cooperation systems with only one slave are not the most efficient potential cooperation structure. So, a more complicated master-slave cooperation

systems might be introduced to get an efficient multi-robot system. These systems consist of one master and a group of slaves dominated by the master. This system has a radial structure. One of the major disadvantages is that if the master makes errors, the whole master-slave system will not work properly. The master is the single point of failure of the master-slave approach. Its work is irreplaceable in the system. This could be overcome by using two radial structure, each of them having one master and a group of slaves. The communication should take place only among the masters to detect potential failures of each other, and to divide the main task between them. Figure 3.2 presents this cooperation structure, master A and master B are the masters of all other slaves dominated by each of them respectively. Both of them cooperate to avoid potential failures, and to increase the system performance of the multi-robot system.

When a potential failure is detected by any one of them e.g. master A, then a correction command should be sent to announce this failure. Sometimes other kind of failure occurs if the robot (e.g. master A) looses the ability of using some of its resources, which is called robot malfunction. Detecting this malfunction is easily achieved by other robot (master B) if the master A does not reply to any messages sent by master B. Once a fault is detected, the most obvious solution would be to let master B do the task assigned to master A.

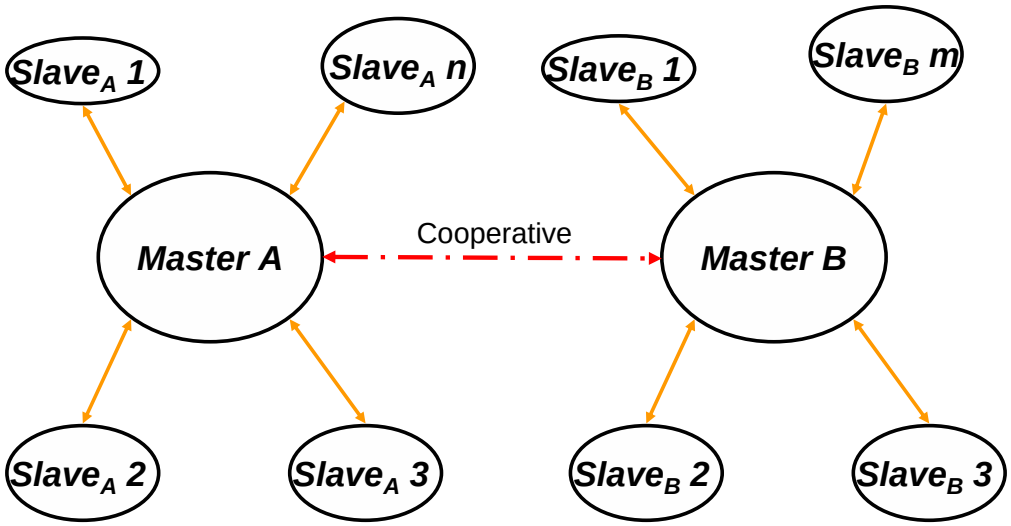


Figure 3.2: Robust master-slave cooperation structure

Within the current work, a master quickly builds 2D map, finding some features to be used in the localization process by the slaves when they are doing their tasks. The master makes a list of different positions in the environment as target places to be scanned by slaves, to get an accurate 3D map. Knowing the positions of its team members, the master sends commands to each of them to do one of these tasks taking into account the conflict solving strategy. A slave receives such a command, executes it, and finally reports the result back to the master. In this type of cooperation concept, the slaves seem to lose their autonomy, where the autonomy is an important property of the system requirement. In fact, the slaves only lose part of autonomy, i.e. they still have some autonomy while planning their path to the target. Additionally, they have their own styles and manners to perform the scanning and to build the 3D map of the target position chosen by the master.

The major advantage of this cooperation concept is the benefit from the master-centred relationship. The master takes care of all its slaves, providing them with tasks, solving their conflicts, and receiving their local maps, and fusing those maps into a global map. So, the slaves within this system can be efficiently coordinated. In this way, the robots will not spend time on negotiating with each other to solve conflicts. This is the reason for the second advantage of this structure namely that the communication cost on the coordination is low. This leads to a higher efficiency. However, this structure has the disadvantage of losing some degree of autonomy, and the failures of one the masters need to be detected correctly by the other.

3.3 The Autonomous Mobile Robots

This section presents the mobile robots that were used in the course of this work. The main structure of the mobile platform Odete2 is presented in subsection 3.3.1, its software control architecture in subsection 3.3.2. In subsection 3.3.3 the master and slave robots are introduced in more detail, presenting what type of sensors they have, what tasks they will carry out, and what type of information can be exchanged between them.

3.3.1 Mobile Robot Odete2

Figure 3.3 shows the platform Odete2, which was developed at the chair for Industrial Applications of Computer Science and Micro Systems (IAIM) at the University of Karlsruhe.

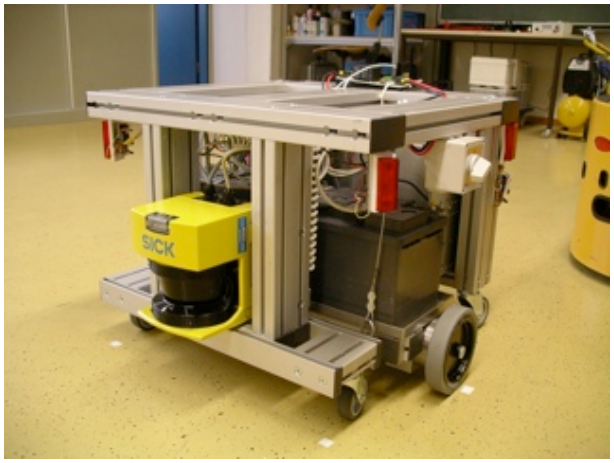


Figure 3.3: Mobile robot (Odete2)

This vehicle is equipped with two driven wheels (differential drive) including shaft encoders for motion tracking. The robot is able to move at a speed of up to $1.2m/s$. Four caster wheels are integrated additionally for keeping the robot upright. A planar laser scanner is used for the navigation process through the environment. It is at the same time a component of the security concept, since it is necessary to detect any obstacles. Detecting an obstacle in the protection region of the laser scanner or pressing one of the emergency stop buttons

results in an immediate stop. Being equipped with two long lasting batteries, the robot is able to move independently for up to eight hours without interruption.

Figure 3.4 gives an overview of the control concept and hardware components of the Odete2. The main control is based on an embedded PC. The connection and communication to the periphery take place via PC104 plug-in modules. Sensor information from incremental encoder and laser scanner are conveyed over a CAN bus and a RS422 module respectively. A PC104 D/A converter plate gives the desired number of revolutions to the motor controllers. Moreover, a digital I/O modules takes over the querying of emergency push buttons and the state of the laser scanner. The embedded PC has a built-in Ethernet connection. However for the wireless communication, an additional PC104 module is necessary for the admission of a PCMCIA radio card.

3.3.2 Modular Controller Architecture (MCA)

To control the robot with its sensors, a Modular Controller Architecture (MCA) as developed by Scholl and his group [Scholl 00a], [Scholl 01] and [Scholl 00b] is used. With the help of this architecture it is possible to put new robot prototypes into operation very quickly and to test them without having any further implementation work to do. The latter is only necessary if different mechanical aspects or new sensor components have to be integrated.

The creation of modules with standardized interfaces has the advantage that these modules can be reused in other projects. Therefore, the expenditure of the implementation and of the verification process are clearly reduced. Communication between and synchronization of the different parts of the software has to be done by the controller system, so the project developers can focus their work on the methods necessary for controlling the robot. MCA is to be realized on real time processing systems such as real-time Linux. To program and build the modules required, the C++ programming language was chosen.

A common Graphical User Interface (MCAGUI) was developed within this framework in order to provide a flexible method to control actors and to visualize sensor information. Even 3D illustrations of robots are implemented. Furthermore, mobile robots can be controlled and observed via wireless Ethernet. In the same manner, the internal module parameters can also be changed via Ethernet. That means different parameter settings can easily be tested and compared without recompiling or restarting of the system. This is done by using the graphical browsing utility named MCAadmin, which is a powerful tool to administrate the MCA.

3.3.3 Master and Slave Robots

Figure 3.5(a) presents the Odete2 platform equipped with a colour Bumblebee stereo camera from Point Grey Research at 640x480 image resolution. It is a firewire camera and can capture colour stereo images with up to 30 frames per second. This platform is to be used as the master in the course of this work.

The master tries to extract some natural landmarks based on the stereo vision sensor, and it tries to build the feature-based map to be used in the localization and navigation of the

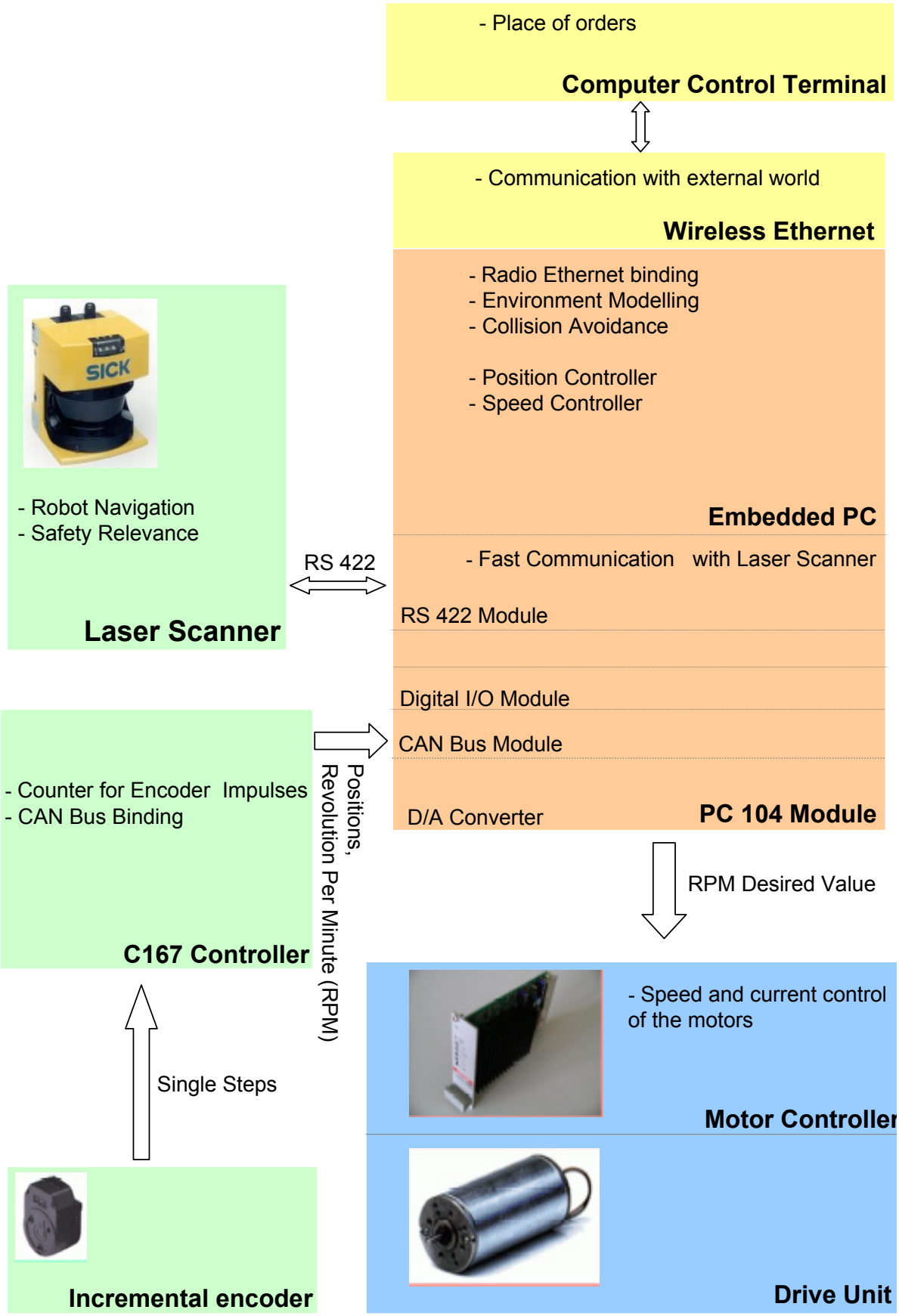
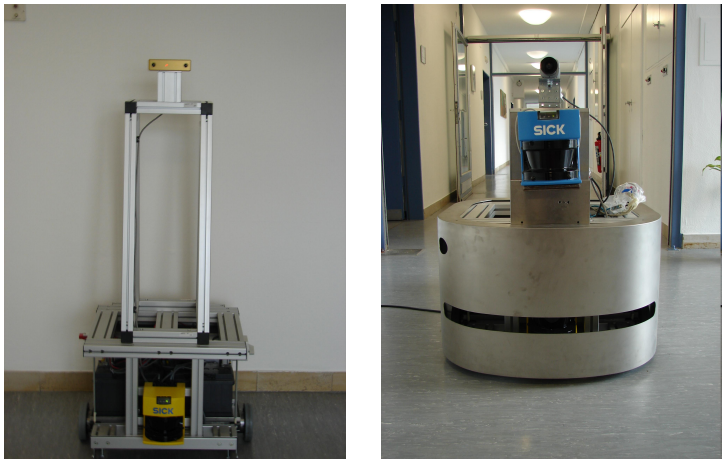


Figure 3.4: Arrangement and interaction of the different hardware components of the Odete2

robots. Moreover, the master has the responsibility to coordinate among the slaves to avoid the conflicting tasks.

Each Odete2 robot has a planar laser scanner. Besides collision avoidance, this sensor is to be used by the master to build up the 2D occupancy map. This map will be used in the process of the path planning and to define the area to be explored by the master.



(a) (b)

Figure 3.5: Mobile robots: (a) master mobile robot and (b) slave mobile robot

The master also has the responsibility to direct the slaves to different locations in the environment to get a 3D map of this area. Thus, a communication has to take place among the robots, and different types of maps, the position of the robots and the target place chosen by the master are interchanged among them.

A new sensor geometry was developed at the chair for Industrial Applications of Computer Science and Micro Systems (IAIM) at the University of Karlsruhe. It permits a conventional 2D laser scanner (SICK LMS 200) to freely rotate around its optical axis, and is named the Rotating Sick laser scanner RoSi [Steinhaus 03]. Its assembly on a mobile platform Odete2 is shown in figure 3.5(b). With the sick LMS 200 scanner a spherical cone with an opening angle of approximately 100° in front of the vehicle is captured.

This platform structure is to be used by the slaves. The slaves use the feature-based map built by the master in order to localize and estimate their position in the environment. By using the occupancy map, they plan a path to the target position chosen by the master, in order to get a 3D map of that location. Based on their sensor (RoSi), the slaves capture the data in the form of range images. This type of data is to be sent to the master. Then, the master fuses different range images captured by different slaves and integrates them into the overall environment model.

As a general result, the master and slave robots have different sensor capabilities depending on their tasks. A quick formation of a 2D feature-based map is needed, for that reason, a stereo camera sensor is used which can capture images up to 30 frames per second. Therefore a task is achieved in efficient time. On the other hand, a 3D model of the certain places in the environment are required, beacuse of that, a RoSi scanner sensor is used without taking the scanning time into consideration.

3.4 Environmental Representation

Research on mobile robot navigation has resulted in different paradigms for mapping indoor environments as was presented in section 2.2. Two examples of these paradigms are feature-based approaches and grid-based approaches. Both approaches to robot mapping exhibit orthogonal strengths and weaknesses. Grid-based approaches have been proven to be very simple and quite useful for obstacle avoidance and planning purposes [Elfes 87]. Their second advantage is that, without going into specific details of the structure of the environment, they can describe the free space quite well, which is useful for applying the exploration strategy. However, the computational, storage, and communication costs of exploration using grid-based approaches scales poorly with time.

Feature-based maps, on the other hand, require significantly less memory than grid-based approaches, and therefore scale better with the size of the environment. Furthermore, they provide an elegant way such as extended kalman filter-based approaches to manage the uncertainty of the localization. This is especially true in the case of position tracking. However, these approaches suffer from the data association problem. For example, if the robot traverses two places that look alike, the feature-based approaches often have difficulties determining if these places are the same or not.

As a conclusion, the feature-based map and the grid-based map are to a certain extent complimentary. The former is superior for long-term localization and the latter is more suitable for path planning and free-space visualization.

Thus, through out this thesis, both types of maps are used to model the environment in order to benefit from the advantages of each approach. Each map type is independently updated. For example, in the case of the master, by using the laser scanner sensor the occupancy grid map is updated, which is used for path planning and free space visualization. Whereas by using the stereo camera, the landmarks are extracted and the feature map is updated, which is used in the localization process.

3.4.1 Grid-Based Maps

One of the most popular space representation models are occupancy grids. Occupancy grids are discredited random fields where the probability of occupancy of each independent cell is maintained [Elfes 87][Moravec 88]. They have been extensively used in robotics mainly due to their simplicity and suitability for decision-theoretic approaches. Some recent examples of their applications are [Pandey 07] and [Rocha 05]. This type of maps can represent any kind of environment. This is done by representing a horizontal 2D space in the environment as a two-dimensional array of cells. Each cell has a certain probability of being occupied. Furthermore, the quality of the map can be adjusted by adapting the resolution of grid cells. Figure 3.6 shows an example of such an occupancy grid map with a grid cell resolution of $5cm$ and an overall environment dimension of $30 \times 30meter$. In this figure, the white cells visualize the free space, the black cells visualize the obstacle states, and the grey cells visualize cells with an undefined state.

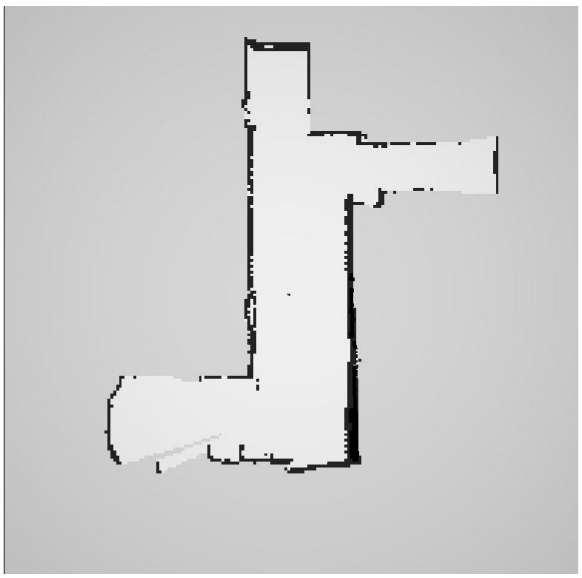


Figure 3.6: An example of an occupancy grid map

3.4.2 Feature-Based Maps

Feature-based approaches such as demonstrated by Rencken [Rencken 94] works by locating features in the environment, localizing them, and then using them as known landmarks by which to localize the robot as it searches for other landmarks.

Artificial and natural landmarks-based robot localization is one of the most popular and well studied approaches for indoor autonomous mobile robot navigation. Artificial landmarks are structured objects specifically placed in the environment to aid localization and navigation and can be detected using image processing algorithms such as edge detection, region segmentation, and object recognition. Examples of artificial landmarks include barcodes, colour-coded signs, and reflecting plates.

Figure 3.7: An example of a natural feature

Natural landmarks are objects or features already present in the environment and serving some other primary function besides aiding in robot localization and navigation. These can be structures in terms of their shapes (e.g. doors and walls), and the position of these features is represented by Cartesian coordinates. Figure 3.7 shows a door as an example of such feature. Its position, the top upper corners of this door, the frame of the door, and the doorknob represent the characteristics of this feature. Such natural landmarks can be detected relatively easily using a variety of image processing techniques as in [Rous 05], [Huwedi 06] and [Lee 04].

3.5 The Framework Architecture

3.5.1 Concept Definition

Many issues must be considered when designing a multi-robot system, such as autonomy, cooperation, communication structure and coordination. Collective autonomy refers to the ability of robots to work individually and without human intervention. Cooperation is the ability of robots to work with each other. Robot cooperation requires a certain amount of communication whenever robots' actions depend critically on knowledge that is possessed by other robots. Coordination addresses the interdependency management among the cooperative robots to achieve individual or collective goals.

All of these issues are addressed using a new framework architecture designed in the course of this work. Figure 3.8 shows the architecture developed for 3D mapping in multi-master/slave environments based on these issues. It consists of two main blocks. The first block demonstrates the work done by a master; and the second demonstrates the work done by a slave. As presented in section 3.2, a hierarchical master-slave approach is used to coordinate the robots in order to let the master solve any work conflict that may happen. The coordination is done between the robots through some communication protocols which were developed in order to maximize the performance and efficiency of this structure.

The design of all modules and the communication protocols were done using the MCA architecture, presented in subsection 3.3.2. The MCA architecture was used in order to facilitate the communication between all parts and modules of the system. In other words, every module is structured to have five communication interfaces, four of them enable a connection to other modules, whereas the fifth one can be used to read and change the internal parameters of a module even while the software is executing. Besides these interfaces, communication interface modules were implemented based on these structure of the aforementioned interfaces, by which certain types of data between two different robots can be exchanged. These types of data are either in the form of a local map of the environment, or in the form of the position of the robots. This interface is implemented in order to achieve the coordination task during the use of the communication bandwidth by the robots in an efficiently.

Within the MCA architecture, different parts of our framework are designed as a basic building unit of the MCA, called a module. This concept allows to develop each part of our framework independently in a safe environment, fix all the bugs, and then connect all modules together to perform the complete work done by either the master or by the slave. There are a number of important benefits from this structure:

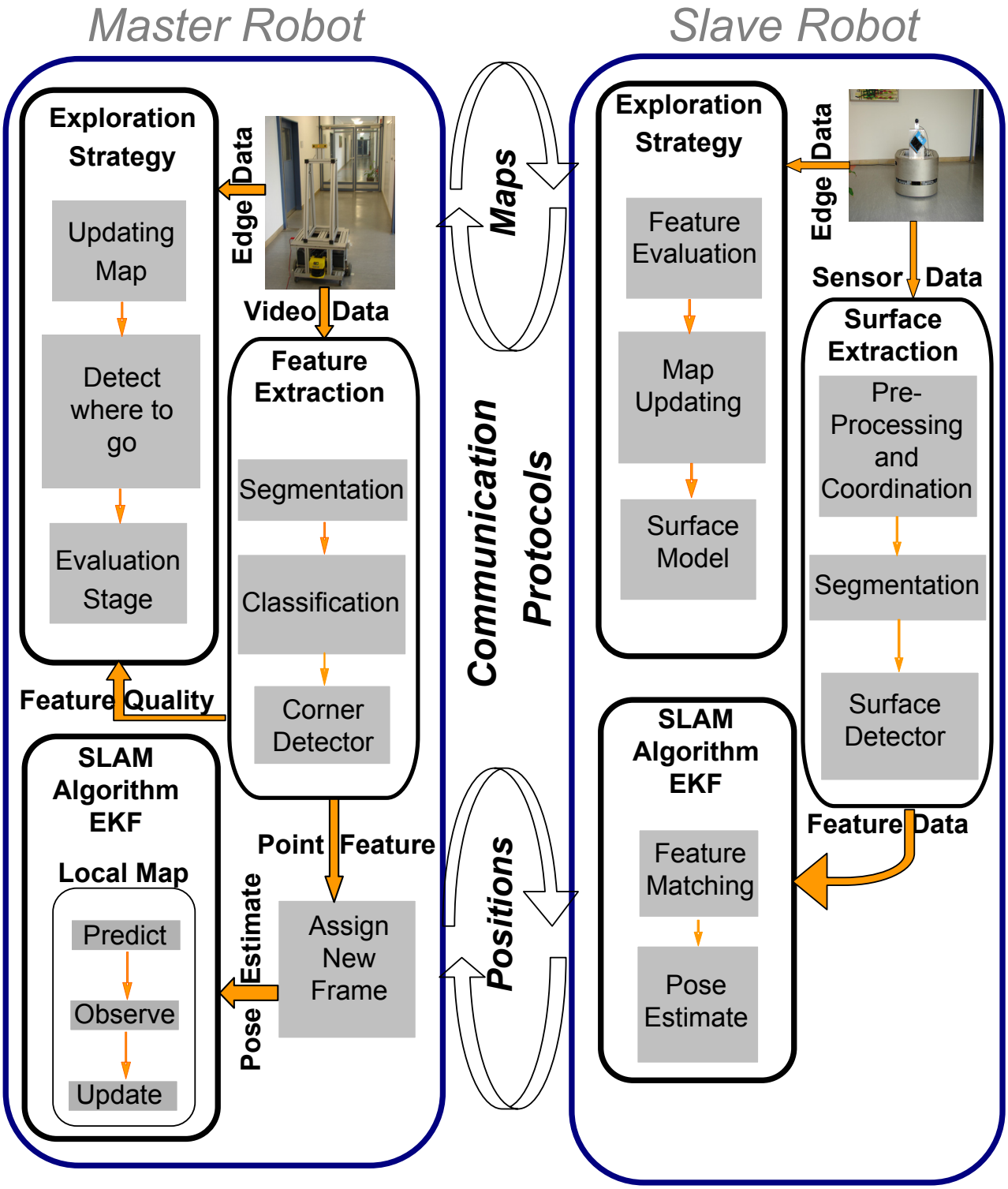


Figure 3.8: The framework architecture for 3D exploration using multi mobile robots

1. To perform complex functions in an efficient way. Complex functions are like sensor interface processing, behaviour control and providing different interfaces to different sensor capabilities.

2. The handling of different types of data, produced by cameras and laser scanner between different models of the framework.

3. The reusability of the modules in different algorithms by different robots in order to achieve an efficient development and realization of this framework.

3.5.2 Algorithms and Modules of Master/Slave Robots

Our framework consists of cooperating algorithms that make up a reusable design for specific software modules. The framework architecture defines how components are integrated into the framework and how they are interrelated. Each block of this framework consists of three main algorithms, performing different tasks in both cases (master/slave robot). These are the exploration strategy algorithm, the feature extraction algorithm and the SLAM algorithm. To achieve efficient algorithms, some kind of data has to be interchanged among the robots.

- Exploration strategy algorithm:

  In the case of the master, the edge data from the laser scanner is obtained. This data is applied to the exploration algorithm, which tries to build and update the grid-based map, to detect where to go next based on some evaluation steps. Additionally, input from the feature extraction algorithm is necessary to complete the evaluation steps. It is known as the feature quality in the surroundings, which is applied in the exploration algorithm to evaluate the destination to be selected. In the case of the slave, the edge data from the laser scanner is captured and compared with the map obtained from the master. A target from a list of destinations generated by the master is chosen based on some prospect values. A path to this target is then planned and the new place is modelled as a 3D representation. More information and a description of this algorithm are presented in chapters 4 and 6.

- Feature extraction algorithm:

  In the case of the master, video data is captured from the stereo camera. This data is applied to the feature extraction algorithm which finds interesting features in the environment. These features are used by the slave for the localization process. The master generates a point feature from the output of the feature extraction algorithm to be used in the SLAM algorithm. The slave however, needs some pre-processing and coordination before it starts the process of feature extraction. By using the RoSi scanner sensor the data is captured. Then, the segmentation and surface detector is performed in order to find the feature detected by the master. The feature output will be used in the SLAM algorithm to match it and to estimate the pose of the robot. More information on this process is presented in chapter 6.

- SLAM algorithm:

The local feature-based map is built by the master. It predicts its position, observes a new measurement, and then updates the map. This process is repeated as long as new data is coming from other algorithms. The algorithm uses the extended kalman filter to perform the SLAM process. The resulting map is sent to the slaves. They try to match these landmarks with the features extracted by their respective sensors, and then estimate their pose in order to get an accurate 3D model of the different destinations chosen by the master. A more detailed explanation of this algorithm is presented in chapter 4.

3.6 Conclusion

This chapter presented the concept of our work for 3D exploration using multi mobile robots. It started with giving an overview about the limitations of using a multi robot system, and discussed how these limitation can be avoided by introducing the master-slave framework.

The second section introduced the basic definition of the master-slave structure, in particular concerning the cooperation between master and slaves. The advantages and disadvantages of this structure are also presented, showing that a failure done by any master can be detected correctly by the other, and using their cooperation to cover its task.

The third section gave a closer look at the mobile robots. The main structure of our platform Odete2 was presented, also showing its software control architecture. The hardware components of the master and slave robots were demonstrated, their sensor types and the tasks to be accomplished were presented. The discussion also included the type of information that can be interchanged between them.

In the fourth section, the environmental representation used within the work was presented with their feature-based and grid-based maps. To a certain extent, both maps compliment the advantages of each other. The feature-based map is superior for long term localization, whereas the grid-based map is more suitable for path planning and free space visualization.

In the fifth section, the framework architecture for 3D exploration using multi mobile robots was explained, showing the advantages of this concept. This architecture defines how the components are integrated into the main framework and how they are interrelated. Also, which algorithms are performed by the master and by the slaves were illustrated in details.

Chapter 4

Exploration Strategy Algorithm

4.1 Introduction

Active exploration is the problem of controlling a robot to decide on the basis of the current map where to go to in order to maximize its knowledge about the external world as well as to improve the current map. When a robot or a team of robots explore an unknown environment and build a map, an exploration strategy is needed to acquire as much new information as possible with each sensing cycle, so as to minimize the required time to completely explore the environment. The goal of any exploration strategy is to answer the question "*where should the robot(s) move to?*", i.e. robotic exploration is the task of generating the robot's motion in the pursuit of building a map as fast as possible.

Exploration is a challenging problem because robots have to cope with partial and incomplete models, and in the case of multi-robot exploration, robots also have to coordinate their motions in order to reduce their mutual interference and to take the advantage of simultaneous operation by simultaneously sensing different regions of the map. Some quantities that are usually traded off are the navigation cost, the possible loss of pose information along the way, and the expected gain of the information. A correct choice based on these prospect values would improve the performance of the strategy employed. Unlike traditional path planning where robots make decisions to get to a specific destination, a mission of the exploration strategy is to make decisions that help them collect information from varied destination areas in an efficient way. The robots do not have a destination to reach, but rather tasks to complete.

Within the current work, the primary focus is on the problem of using an optimal exploration strategy in order to cover the whole environment in a minimal amount of time. In particular, we develop a formalism for implementing the exploration strategy that optimizes criteria such as navigation cost, localizability, and uncertainty reduction. A method to extract features from sensor data during the course of this process is employed and to fuse these features into a common global map. The result is a robot that autonomously explores its environment, optimizing the exploration at each stage and merging newly acquired sensor data into its existing map. Figure 4.1 shows the structure concept of such a system, which acquires a 3D world model using different sensors. It consists of different processes, starting with data

capture from various sensors, followed by integration of the data into navigation and path planning. The sequences are repeated until the given task is achieved.

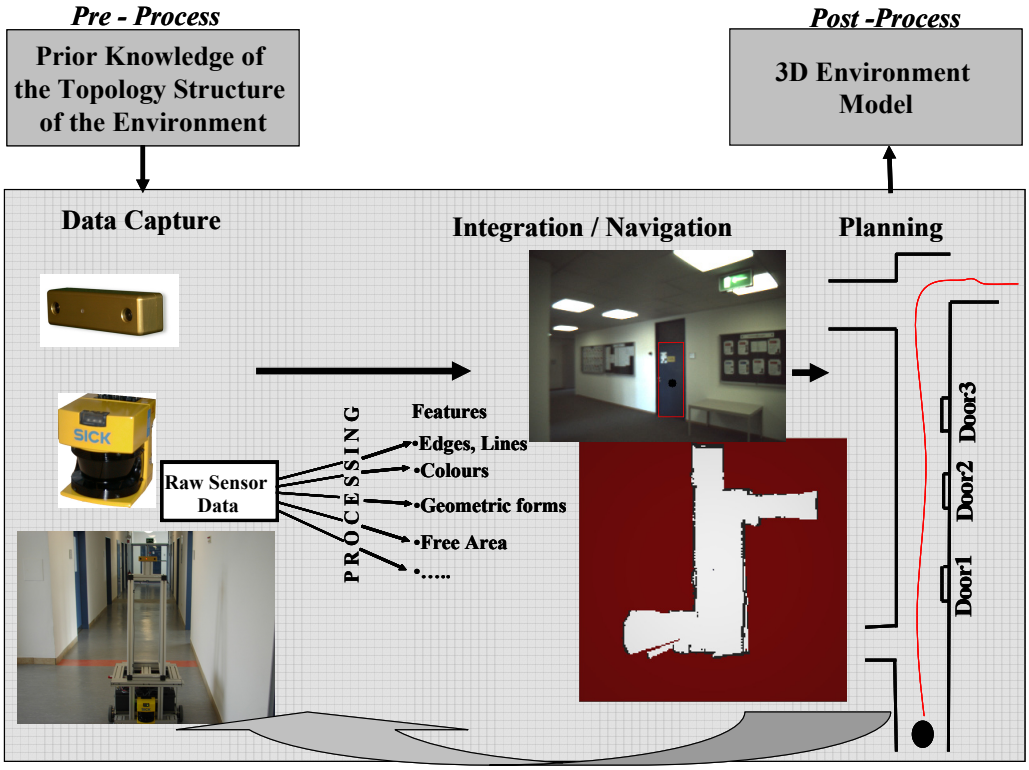


Figure 4.1: The concept of autonomous 3D feature-based exploration using different sensors

To achieve the goal of the exploration process our intended prospect value functions is constructed in such a way that it balances the desire to see as much formerly unknown environment as possible with the need to have enough overlap with and landmark information of the already scanned part of the indoor environment to guarantee a good map registration and robot localizability. One of the map representations used are grid-based maps. These maps are captured from the data of the laser range finder sensor. While exploring an unknown environment, we are especially interested in the areas which lie on the edge of the explored and unexplored regions. These are known as frontier regions established by Yamauchi [Yamauchi 97]. If a robot is directed to such a region, we can expect that it gains more information about the environment. The fact is that a map generally contains several frontier regions. So to achieve the goal of the best exploration time the robot must choose one of them in order to get more information about the environment, to keep the necessary exploration time short and to find some features to improve the location process. In the next section, we describe how we compute the cost of reaching frontier regions and how the startegy developed in the course of this work is operating.

Two different sensor capabilities are used to achieve the task of the exploration strategy. The stereo camera sensor is used to extract and recognize landmarks around the robot to improve the localization process during the strategy process. The landmark extractor algorithm uses a region growing approach which is presented in chapter 6. A laser range finder sensor is used to capture the occupancy grid map, in which free space, obstacles, and areas that are still

to be explored are represented. The final result of the exploration is a multi-representation map consisting of grid values and the position of the landmarks.

The outline of this chapter is as follows: In section 2 we introduce some prospect value functions used in the strategy, discuss their theoretical aspects and how they are used. In section 3 our algorithm is presented for both the single-robot and the multi-robot case. The experimental results concerning the exploration strategy are demonstrated in section 4. Some final conclusions are drawn in section 5.

4.2 Prospect Values of Regions of Interest

The prospect values of a region of interest indicates how much new information can be gained by exploring this region, declaring the properties of this region which could be useful in the process of exploration. In order to select one target from a list of several interest regions, the algorithm computes the prospect values of each region and chooses a target that has the best evaluation values.

Within this section, we define the idea of a frontier region and its usage to get more information about unknown places. Then different prospect values are computed for each region which might be selected as a target of the next move. The navigation cost as the first prospect is based on the distance travelled by the robot to the target region. It represents the goal that a long travel time should be avoided to improve the efficiency of the exploration. The next prospect is based on the availability of the features at that target. As a result of collecting data with a stereo camera sensor and applying the object detector algorithm during the process of a 360° sensor sweep, a number of features are detected. A patch of information about each feature is built. This patch describes the visibility of this feature and its relative position. Then a region with a high feature visibility should be selected to improve the localizability. The third prospect is used in the case of multi-robot systems in order to prevent conflicting tasks and sensor interference problems. At the end, the total prospect of each target point is calculated in order to choose one of several regions of interest.

To present the idea of each prospect value, a grid map with a simple geometry is used. This map is applied to the algorithm to test the validity of the procedure.

4.2.1 Defining the Prospects of Frontier Regions

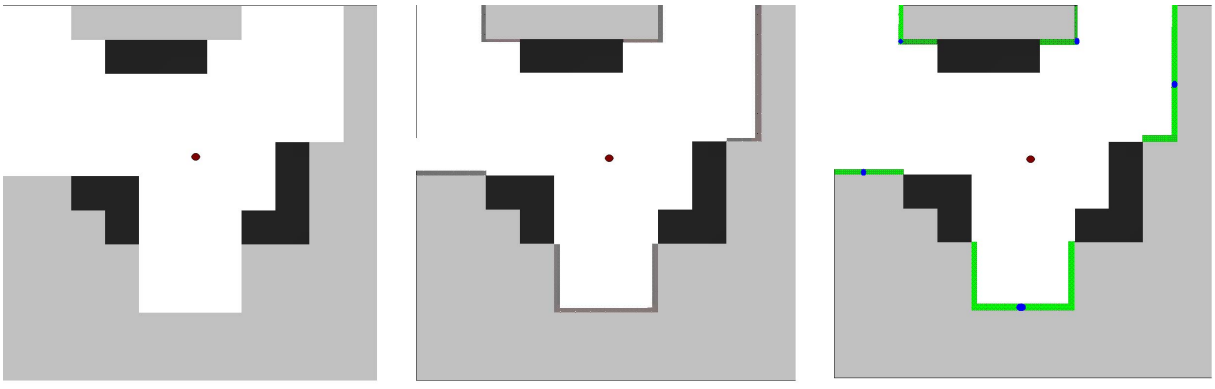
The central question in exploration is that "*Given what you know about the world, where should you move to gain as much new information as possible?*". Yamauchi in [Yamauchi 97] and [Yamauchi 98] answered this question by developing a frontier-based robot explorer designed to explore complex environments typically found in office buildings. The premise of this method is that "*To gain the most information about the world, move to boundary between open space and uncharted territory*" [Yamauchi 97]. In this method, an occupancy map is used as the environment's spatial representation. This map consists of cells, in which the probability is stored that the corresponding region in space is occupied. Each time a sensor reading is obtained from the robot's laser scanner, a sensor model is used to update these cells.

Initially all of the cells are set to the prior probability of occupancy, which is a rough estimate of the overall probability that any given location will be occupied. To detect frontiers, the occupancy probability is tested based on the threshold of prior probability, and each cell is placed into one of three classes:

- **Open:** cells with an occupancy probability < prior probability
- **Unknown:** cells with an occupancy probability = prior probability
- **Occupied:** cells with an occupancy probability > prior probability

where in figure 4.2, the open cells are represented by a white color, the unknown cells are represented by a grey color, and the occupied cells are represented by a black color.

A process analogous to edge detection and region extraction in computer vision is used to find the boundaries between open space and unknown space. Figure 4.2(a) shows a simplified example of frontier edge detection. Any open cells adjacent to an unknown cell are then labelled as a frontier edge cell. Figure 4.2(b) presents all frontier cells that were found in this map. In addition, adjacent edge cells are grouped into frontier regions using an image segmentation technique known as region growing. Any frontier region above a certain minimum size is considered as a frontier region and added to a list of the destinations to be explored. The minimum size is related with the size of the robot to allow the robot to pass through it. Figure 4.2(c) presents the result of finding several frontier regions, showing the centroid of each region as a potential target point of the next move of the robot.



(a) (b) (c)

Figure 4.2: Frontier-based exploration: (a) simplified example of the used grid-based map, (b) results of finding frontier cells and (c) results of finding frontier regions

4.2.2 Prospect Value of Navigation Cost

Figure 4.2(c) presents different target frontier regions to be explored. To determine the cost of reaching each frontier cell, we compute the optimal path from the current position of the robot to all frontier cells based on the navigation function of algorithms developed in [Latombe 91] and the distance transform algorithm used by Zelinsky [Zelinsky 92]. In our

approach, the cost of driving from the current position to the proposed destination is based on the information in the occupancy map. To start computing the minimum cost path, the following initialization step is needed:

1. Initialization. The grid cell that contains the robot location is initialized with 0, all other with ∞:

$$E_{x,y} \leftarrow \begin{cases} 0 & if \ < x, y > \ is \ the \ robot \ position \\ \infty & , otherwise \end{cases} \tag{4.1}$$

where $E_{x,y}$ is the grid cell value at a position of x and y.

After setting the cells to initial values, the program repeatedly scans through the grid and tries to reset cells to lower values. The lower cost is computed by looking at the neighbouring cells and using an estimate of travel cost from the adjacent cells to the current cell. For laterally adjacent cells the estimate is 1.0. For diagonally adjacent cells the estimate is a square root of two times the distance of the laterally adjacent cell. The distance transform wave emanates from the zero cell (robot position) and travels uphill until it encounters an obstacle and has no where else to go. Thus, the number in each cell is an estimate of the cost of travelling the robot from its position to the position of this cell.

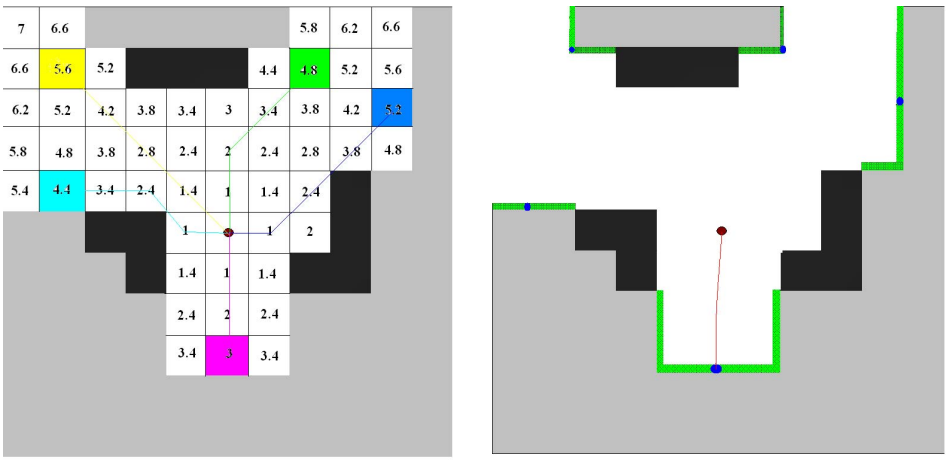
The net effect is that following a steepest descent path through the distance transform from any frontier cell will lead to the path travelling cost. Figure 4.3(a) shows an example of applying the distance transform algorithm to the map from figure 4.2. To find a path to each frontier region, locate the centroid of this region in a grid cell and then descend downhill starting with the neighbour with the least distance transform. While calculating the path it sometimes happens that two adjacent cells are with the same distance value. In order to minimize the number of turns, the direction that is chosen is the one that the robot is currently facing if this is possible.

The cost of reaching any frontier region is calculated, the navigation cost of all destinations is sorted in ascending order of their distance, and the prospect value for each destination is calculated as one utility:

$$E^d_{F_i} = E_{x,y} \tag{4.2}$$

where $E^d_{F_i}$ is the prospect value at the frontier region F_i, d is only an indication that this prospect is a navigation cost function, and $E_{x,y}$ is the grid cell value at the centroid of the frontier region F_i.

Figure 4.3(b) shows an example with a path from the robot position to the best prospect value position of the frontier region in order to continue exploration. Notice that the algorithm produces the navigation cost for all destinations at once from the current starting point.



(a) (b)

Figure 4.3: Prospect value of navigation: (a) the distance transform applied on a regular sized grid and (b) the best path is chosen with a prospect value of navigation cost = 3

4.2.3 Prospect Value of Localizability

The main goal of an exploration strategy is to collect information about the world, while also reducing the uncertainty in the robot localization in order to integrate this information properly into the map. The localization prospect value is used to distinguish between destinations with different localization quality. Conceptually, the quality of localization at any given frontier region is determined by the number of already observed features and their visibility at each destination.

For localization purposes, natural features are extracted from the surroundings by using a stereo camera onboard. The feature extractor algorithm is presented in more detail in chapter 6. In general, the algorithm identifies different types of features. In the office environment that was used for evaluation purposes during the course of this work, these are in particular the doors and fire extinguishers as prominent features of the environment. The algorithm produces for each feature the visibility ratio to be used in the evaluation process. As a result, the features extracted from the surroundings are collected in sets which represent the features available at each frontier region, for example:

$$F_i = \{f_j, vis_j, j = 1, ..., N\},$$

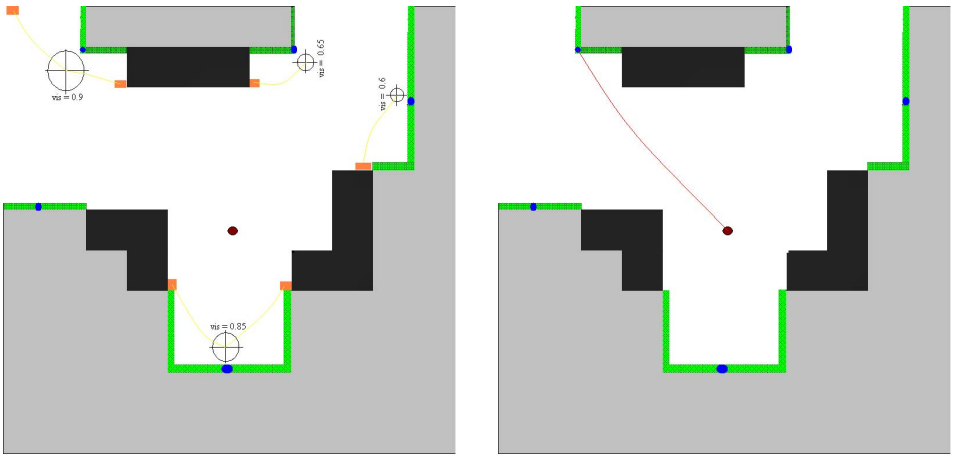
where f_j is a Cartesian representation of the j^{th} feature and vis_j is its visibility appearance. This set is the information of the features available at the i^{th} frontier region F_i.

Once new frontier regions have been generated and the evaluation of the navigation cost is completed, they are evaluated under the view point of localizability, i.e. an estimate of the possibility each of them offers of localizing the robot. In the view of the feature extractor algorithm, it is natural to relate this possibility to the number of features available at each frontier region that will be observable from that position, and hence results in associated pairs. In particular, denoting this number by f_l, the localizability prospect computes the localization cost at frontier region F_i as:

$$E^l_{F_i} = \frac{1}{\sum_{j=1}^{f_l} vis_j}, \tag{4.3}$$

where vis_j represents the visibility of the j^{th} feature. A higher value of vis_j is assigned to the nearest features, whereas a lower value is assigned to the more distant features which are intrinsically more uncertain.

Figure 4.4(a) shows different situations of the localization prospect value for different frontier regions based on the visibility of the features. When the evaluation is validated the robot plans a path to the best region in terms of the visibility factor. Sometimes, the localizability of some region is below a minimum threshold l_{min}, in this case the evaluation process is not validated and the robot tries to move to the nearest region in order to find more features to be extracted and to be used in the localization process. Figure 4.4(b) shows an example with a path from the robot position to the best prospect value position of the frontier region in order to continue exploration.



(a) (b)

Figure 4.4: Prospect value of localizability: (a) calculating the visibility factor at each frontier region and (b) the best path is chosen with a cost of prospect value of visibility = 0.9

4.2.4 Prospect Value of Coordination

In the case of multi-robot exploration, an evaluation of the coordination aspects has to be performed in order to explore the environment efficiently without any conflicts or sensor interference problems. In fact, the actual information that can be gathered by moving the robot to a particular location is impossible to predict, since it depends very much on the structure of the corresponding area. However, if there is already a robot that is moving to a particular frontier region, the prospect value of that region can be expected to be lower for other robots. Furthermore, to prevent the interference problem, the prospect values of the neighbouring regions should be reduced as well, if they are in the vicinity of the robot's target point. This can be measured by using the visibility range of the sensors of each robot.

Within our approach, we propose the prospect value of coordination $E_{F_i}^{cor}$ for each frontier region F_i to be

$$E_{F_i}^{cor} \leftarrow \begin{cases} 0 & if\ it\ is\ chosen\ by\ any\ robot\ or\ in\ the\ sensor\ range\ of\ any\ robot \\ 1 & otherwise \end{cases} \tag{4.4}$$

In order to choose a destination target out of some frontier regions based on the total prospect value of each region, the robot first checks its coordination prospect value. If no robot has chosen this target so far, it selects this region as its target destination. Whenever a frontier region F_i is selected by a robot, the coordination value $E_{F_i}^{cor}$ of this region is set to 0 as indication to all other robots. Furthermore, the coordination factor of the adjacent frontier regions are also set to zero, if they are in the robot's sensor range.

For example, suppose at the stage n of the exploration, there are 5 frontier regions to be explored by two robots. The first one chooses the best region that is not yet chosen by the other. Now, the second robot uses a prospect value of 0 for this region and all regions which are in the visibility range of the first robot's sensors. That means, it will not choose the places in the neighbourhood where the first one is moving to, and it will try to choose an other destination far away from the first robot.

4.2.5 Total Prospect Value of Different Regions of Interest

- **One robot case:** Based on equation 4.2 and equation 4.3, the total prospect of a specific target point $E_{F_i}^{dl}$ is calculated, which is the weighted sum of individual prospects:

$$E_{F_i}^{dl} = \left(E_{F_i}^{d}\right)^{a} + \left(E_{F_i}^{l}\right)^{b}, \tag{4.5}$$

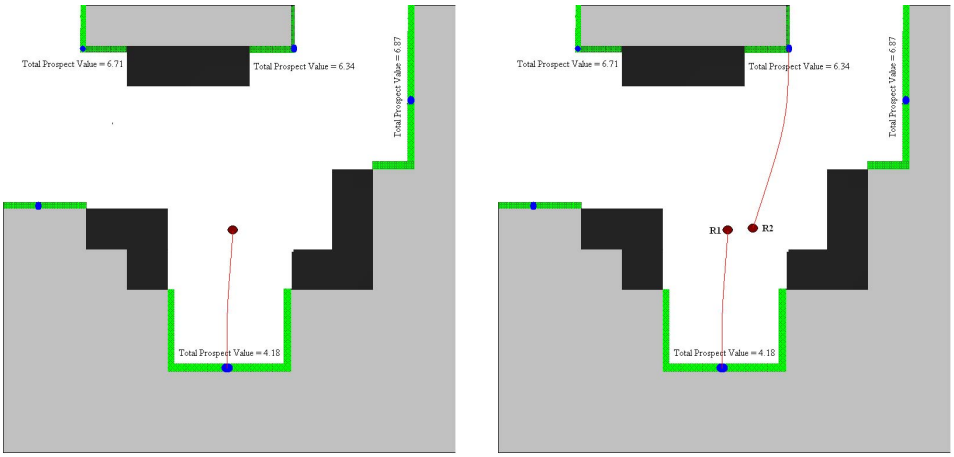
  where the values of a and b are selected depending on the priorities of the respective action, i.e. whether the nearest region is to be explored first or the region that has features with a high visibility factor for a stable localization process. The combined effect of the two prospects can be observed in figure 4.5(a). One frontier region of figure 4.5(a) is deleted from the evaluation step because it has not any features around it, so the evaluation step neglects it and consider it to be with a high value. The experiments presented in section 4.4 were performed with exploring the nearest target that has at least one visible feature in its vicinity. The location with the lowest total prospect is selected as the next destination:

$$E^{*} = \underset{\boldsymbol{F_i}}{\operatorname{argmin}} \left(E_{F_i}^{dl}\right) \tag{4.6}$$

- **Two robots case:** In the case of two robots, the third prospect value is used as well. Based on equation 4.2, equation 4.3 and equation 4.4, the total prospect of any target point $E_{F_i}^{dlcor}$ is calculated as follows:

$$E_{F_i}^{dlcor} = \left[\left(E_{F_i}^{d}\right)^{a} + \left(E_{F_i}^{l}\right)^{b}\right] \bullet E_{F_i}^{cor} \tag{4.7}$$

That means, if the frontier region F_i is selected by one of the robots, then the other one tries to delete this region from its evaluation stage and chooses other regions that are far away from the destination of the first robot. Figure 4.5(b) shows this phenomenon, where the first robot selects the region of the best prospect value, whereas the second robot selects another destination that is far away from the sensor range of the first robot.



(a) (b)

Figure 4.5: Total prospect value of different regions of interest: (a) case of one robot and (b) case of two robots

As a general conclusion, two prospect measures related to the navigation cost and localization quality were described. Combined, they produce a balanced evaluation of the proposed destinations with respect to the overall goals of maximizing map coverage, accuracy and the speed of the exploration. This can be applied in the case of one robot or multi-robots as well. Besides the navigation cost and localization quality prospect values, the coordination prospect was introduced in case of the multi-robot systems.

4.3 Exploration Strategy

The exploration process using multiple robots is actually a complicated task, which requires expertise in different methods from robotic fields such as localization, SLAM, and path planning. To be able to perform exploration, the robots must be able to localize themselves relative to the environment (localization), map the environment (mapping), build a map and simultaneously localize themselves to the map (SLAM), decide on where to go next in order to build the map efficiently (fast exploration and robot coordination). After these steps the robots need to determine how to get to their goal destination (path planning).

This section shows the application of the exploration strategy in different situations. First, it presents a general approach to do exploration for one robot. It then discusses the procedure that is necessary if two robots are used as master of different slaves. Finally, the tasks of the slaves themselves and their approach to solve these tasks are introduced. The strategies

in all these situations use the prospect values presented in section 4.2. They show how the master-slave structure is applied, and how master and slave communicate in order to perform the exploration efficiently.

4.3.1 General Approach

The exploration strategy approach used within our work requires some evaluation steps in order to perform the task of exploring very efficiently and of maintaining the extraction of features which are forming a feature-based map that is used during the SLAM process. The exploration action sequence employed in our work is as follows:

- Perform a 360° sensor sweep to collect more information.
- Determine the set of the frontier regions (potential destinations).
- Determine the features visible from each frontier region.
- Compute for each detected region the evaluation cost E_{F_i} for reaching this region F_i. In each iteration, the mobile robot chooses target point F_i which has the best evaluation $E^*_{F_i}$.
- Plan a path to the selected target using an occupancy grid map.
- Navigate to the target.
- Repeat these steps at each new target, until there are no more regions to be explored.

If the robot is not able to plan or to navigate to the selected target, the next best target is chosen as the next destination. Figure 4.6 presents the sequence flow necessary to achieve the task.

At step 4 of this flowchart, different tasks are to be carried out by the master in order to choose the next destination. These tasks are presented in subsection 4.3.2. Furthermore, when a destination is chosen by the master it plans a path to get there avoiding if possible any path conflicts with other master robot. It then navigates to its target. At this destination, it repeats the sequence in figure 4.6 until there is no region left to be explored. If any failure occurs in the planning or navigation process, the robot tries to plan a new trajectory or to choose an other destination if the current goal can not be reached.

Two different sensor capabilities are used for exploration, and also for updating two different maps independently. A laser scanner sensor is used to cover the occupancy grid map (OG) which contains the frontier area of interest, whereas the stereo vision sensor is used to extract the features around the robot. The feature-based map (FB) is updated to improve the localization process during the strategy process.

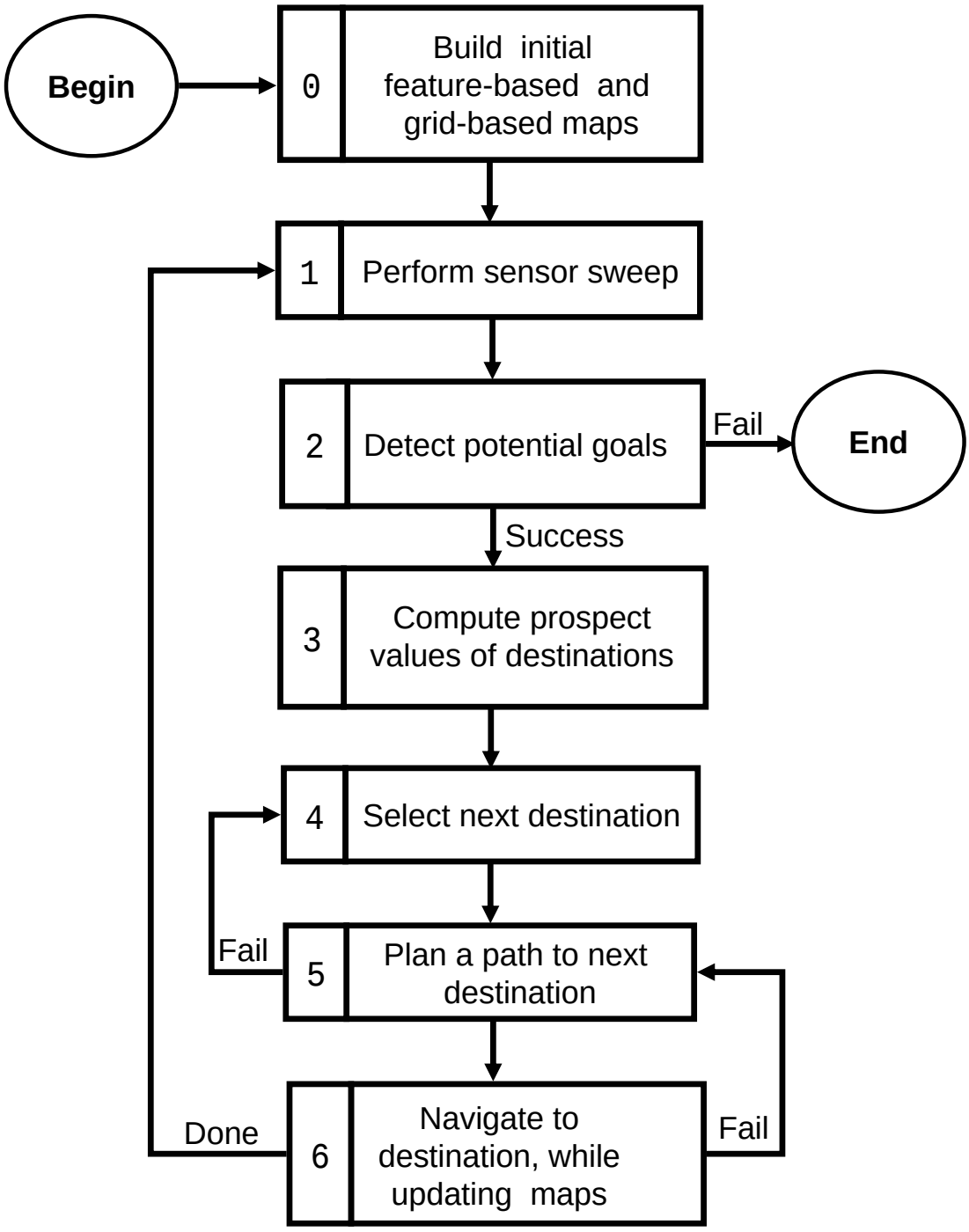


Figure 4.6: Exploration strategy algorithm (general approach)

4.3.2 Master Approach

The master-slave structure of figure 3.2 is to be used in this situation. Two masters are cooperating in order to achieve the exploration task very quickly. Both of them are starting at the same exploration area. Each of them follows the sequence flow presented in figure 4.6. At step 4, they have to select different target destinations to avoid if possible, any work conflict and sensor interference problems, i.e. each one of them should choose its destination far away from the other to get reliable results from both of them.

To perform the cooperation scheme between the two masters, different messages which is presented in tables 5.1 and 5.2, are exchanged. For example, acknowledgement messages are interchanged between them to let the team know the next destination of the other robots. This type of messages allows the robots to coordinate themselves in the environment more efficiently. Additionally, the actual position of the robots are also interchanged between them. By knowing the position of each member of the team, the robots can plan their trajectory such that they avoid any conflicts with the other robot's paths. Furthermore, the local occupancy map is interchanged between them as well. The information in this map is sent in the form of the edges of obstacles in order to use the communication channel in an efficient way. The robots gain information from this map about the existence of unexplored regions. This information is used in the evaluation steps. All of these protocols between the robots give the strategy the power to perform the cooperation in an efficient manner.

Figure 4.7 presents the exploration strategy algorithm adopted in step 4 of figure 4.6. After computing the prospect values of each destination, the robot needs at this moment to select the best of them as the next target destination. In this situation, a prospect value of coordination as described in subsection 4.2.4 must be computed. In order to achieve this, the robot at this stage checks if there is any acknowledgment from the other master. If no messages were received, then the robot selects the best destination as next target and sends an acknowledgment to the other master to be taken into consideration when the other one calculates its next destination.

In contrast, if an acknowledgement was received from the other master, then the robot tries to delete the target of the other master from its own destination list. Simultaneously, it deletes any other destinations from the list that are in the range of the other master's sensor. This strategy distributes the robots evenly over the environment properly to avoid conflict or sensor interference problems.

An other task to be carried out by each master is to direct their team of slaves to appropriate places in the environment in order to obtain a 3D map of certain areas of interest. The tasks conducted by the slaves are presented in subsection 4.3.3. At the last step in figure 4.7, the masters thus send a command to assign a task list to their team of slaves. The command message contains all the information needed by the slaves in order to achieve these tasks.

Three types of protocols are transmitted through the communication bandwidth used by the masters: acknowledgments, robot positions and local maps. It is vital to use the communication channel efficiently, i.e. each robot sends each type of data in a packet. The master easily recognises the type of the received packet, and the size of this packet is limited to achieve as fast a transfer as possible. Factors which influence the efficiency of this strategy

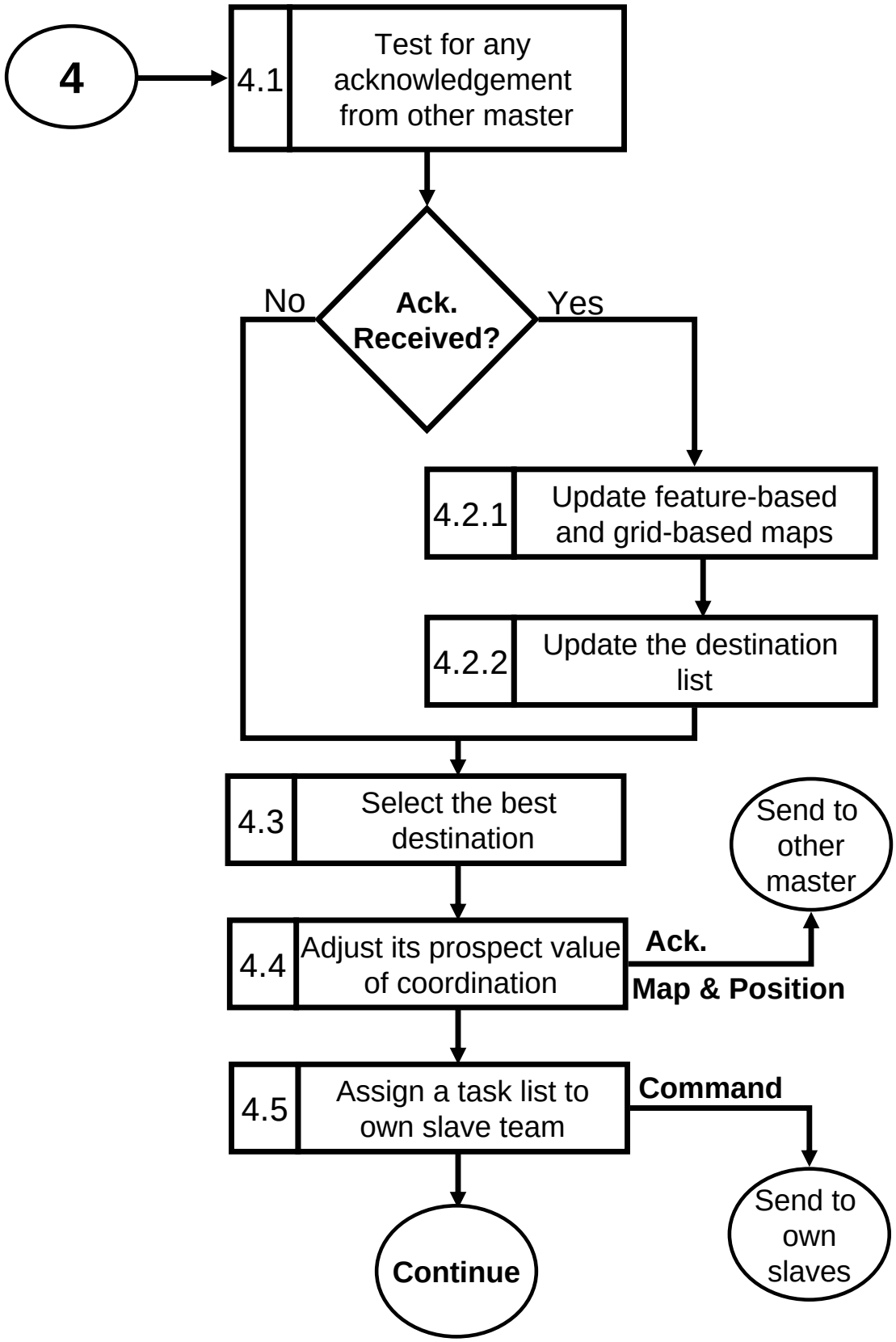


Figure 4.7: Exploration strategy algorithm (master approach)

are how often the channel is used, and how much the communication bandwidth is limited. The performance of this strategy is compared with other strategies in chapter 5.

4.3.3 Slave Approach

In this situation, a simple master-slave cooperation system as shown in figure 3.1 is used. The system consists of one master and a group of slaves dominated by the master. In this system, a slave's tasks are completely determined by its master. The slave receives a command from the master, executes it, and finally reports the result back to the master. In this type of cooperation concept, the slaves seem to lose all of their autonomy, which is an important property of a mobile robot. However, the slaves only lose a part of their autonomy, as they still have a considerable amount of autonomy when conducting the tasks assigned by the master, e.g. by distributing the tasks among them. Within our work, the slaves themselves are cooperating in order to work out the tasks assigned by their master. Figure 4.8 presents the sequence flow of the exploration strategy algorithm carried out by each slave. From this figure, it is clear that the slave team achieves the tasks autonomously in a cooperative manner without any help from the master.

The command packet received from the master contains a list of tasks to be conducted by the slaves. These tasks are of the type of collecting a 3D map of a certain place of interest in the environment. So the list consists of Cartesian values of the destinations of these places. Furthermore, the packet also contains the maps needed by the slaves in order to localize themselves in the environment and to plan a path to these targets. So a copy of the feature-based map and the grid-based map are included in the packet.

Each slave starts working when it receives a command packet from its master. First, it updates its own maps using the newly received information. Then, it performs the evaluation process in order to choose the best target and coordinate itself efficiently with the other slaves. Two of the prospect values presented in section 4.2 are used, navigation cost prospect value and the prospect value of coordination.

The slave applies the rules of best navigation cost by using the distance transform algorithm. This algorithm supports multiple robots and multiple goals. The distance transform algorithm determines the shortest path to the goal from all free-space grid cells. By using this algorithm, the slave chooses the destination of best navigation cost if it is not chosen by other slaves. At the same time, the slave adds other destinations to its local list of targets, those that are in the vicinity of its sensor range. An acknowledgement needs to be sent to all team members of the slave, declaring the destination list of this slave. The other slaves choose other tasks depending on the acknowledgement messages from their team. As a result, the team distributes itself efficiently over the environment avoiding any work conflicts or sensor interference problem. Sometimes, because a slave reserves several tasks for itself, an other slave does not find any task to do. In this case, the master assigns a specific task to a specific slave taking into consideration the prospect values of this task with respect to its slaves.

In terms of autonomy, each slave plans its path by itself, and navigate to its destination. At the target destination, it builds a 3D map using its sensor. The result is sent back to the

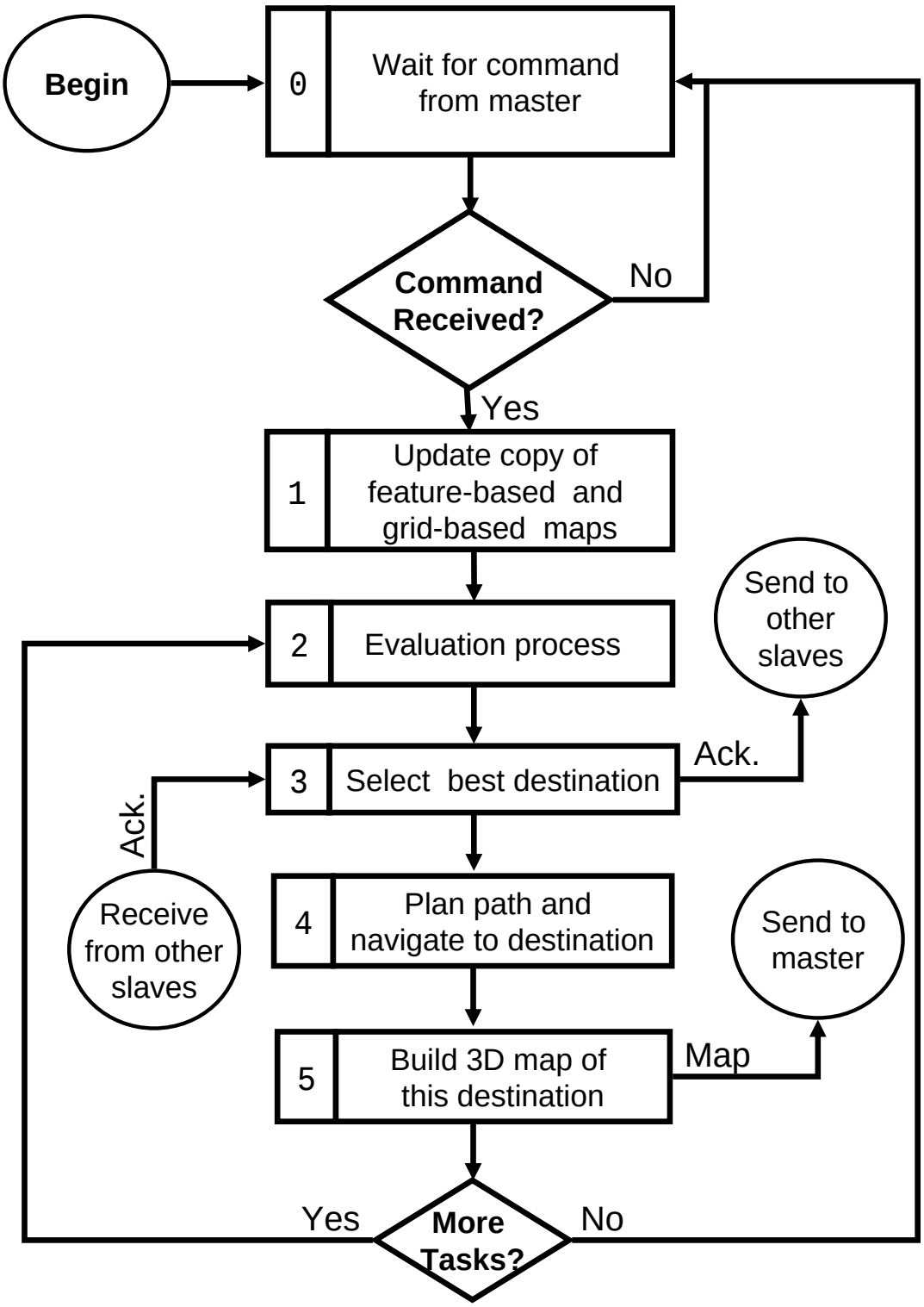


Figure 4.8: Exploration strategy algorithm (slave approach)

master to be fused into the global map. Within this step, a master receives an acknowledge that the slave has carried out the corresponding task. All these steps are repeated by each slave unless there is no task left which is assigned to them.

Apart from the tasks list received from the master, the degree of autonomy of the slaves is relatively high. The slaves select their targets by using some evaluation function, and they perform both path planning and navigation to these destinations autonomously. In the same manner, the degree of cooperation is high, as the slaves distribute themselves over the environment efficiently. This is achieved by interchanging messages among them. However, some drawbacks can be noticed. For example, the evaluation process is repeated by each slave in order to select their targets. Furthermore, due to the amount of interchanged messages, the communication channel can be busy, which sometimes causes a delay. In these cases some slaves have to wait a short period of time in order to get access to the channel to send their messages. Furthermore, the number of slaves in a team and the number of tasks to be carried out by slaves need to be taken into account in order to work out the tasks efficiently. All these factors are studied in more detail in chapter 5.

4.4 Experimental Results

This section presents experimental results obtained using Odete2 platforms equipped with stereo camera and laser scanner sensors as presented in chapter 3. This section consists of three parts. Firstly, it presents experimental results of the exploration strategy algorithm that is described in subsection 4.3.1, using one mobile robot equipped with a stereo camera and laser scanner sensors. Secondly, it presents experimental results of the exploration strategy algorithm in the case of 2 master robots. Within this work, two Odete2 mobile robots are used with the same configuration. Thirdly, the task done by the slaves equipped with the rotated laser scanner is shortly presented, demonstrating only the type of tasks assigned to the slaves. The fusion process of the data obtained by them is evaluated in chapter 6.

Two types of environmental representation are used, and they are acquired independently. The dimension of the applied grid-based map is shown in table 4.1. In the case of the feature-based map, two types of natural landmarks found in the environment are used. These are doors and fire extinguishers. Both of them have their own topological dimension and homogenous colours. The extraction and processing of such features is demonstrated in chapter 6. The total number of the features extracted through out the experiments was ten doors and one fire extinguisher.

4.4.1 Result of the General Approach

The experiments were conducted in an environment of the size of $30 \times 10m$, with a single mobile robot equipped with a stereo camera and laser scanner sensors. Its speed is $0.2m/s$. The robot was only directed through a corridor to present the idea of the algorithm. The robot completed its task in a time of about 8 minutes. The feature-based and occupancy grid-based maps were used to represent the environment. Based on the grid-based representation,

Parameters		Values
Workspace		
Length × Width	(General Approach)	$30m \times 10m$
	(Master Approach)	$20m \times 10m$
Cell Size		$0.20m$
Robot Speed		$0.2m/s$

Table 4.1: Parameters of the robots and the grid-based map in the experiment

the motion planning was performed. The MCAGUI tools of the MCA framework were used to initialize the algorithm and to demonstrate the results of the mapping.

The task of the exploration is achieved in four steps. At each step, the robot applies the sequences demonstrated in figure 4.6. It starts with sensor sweeping to find the features available around it and to detect where to go next, and it ends with navigating to the best target it found.

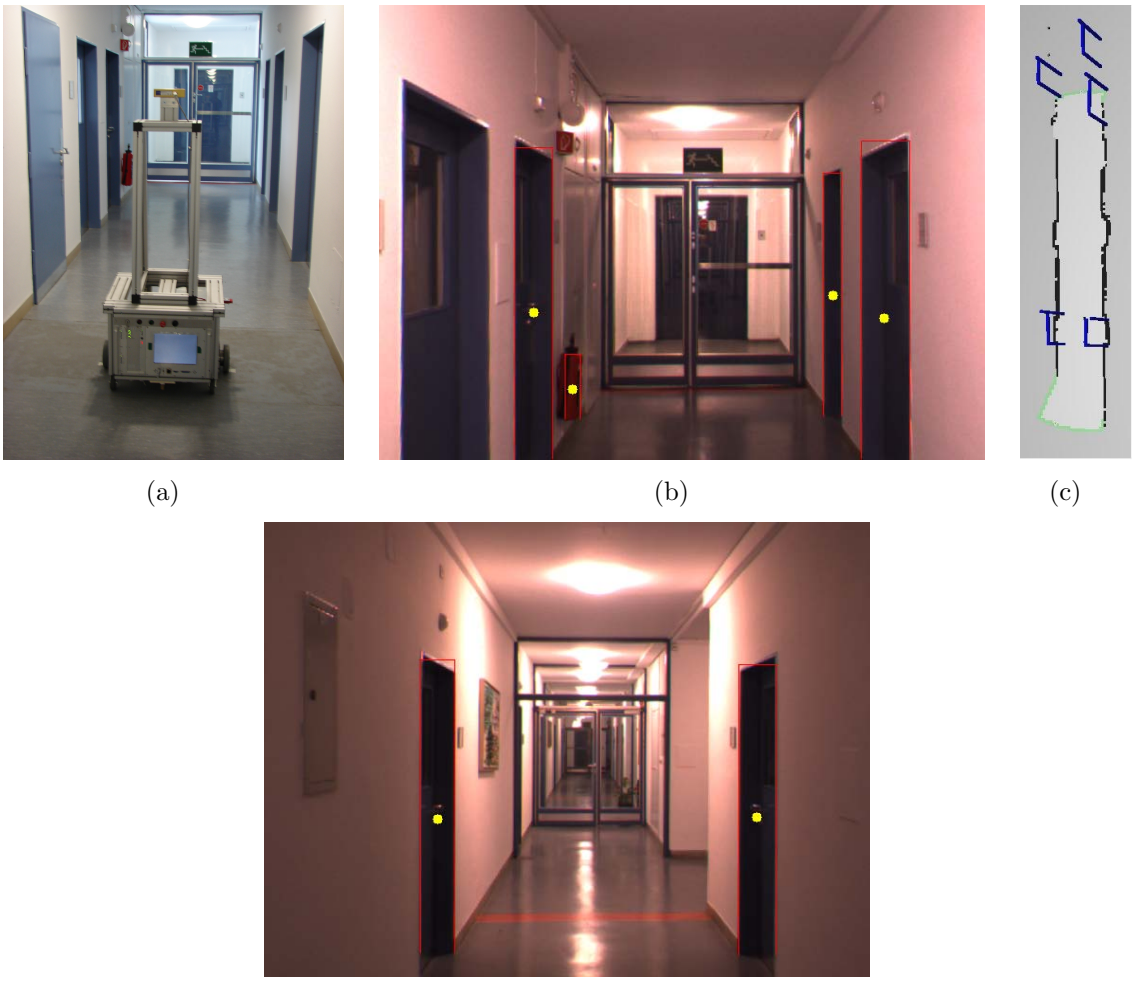
Figure 4.9 shows the result of the first step. Figure 4.9(a) presents the start position of the robot, at which the robot begins to explore. At this stage, the results of the extraction algorithm are found, which are presented in figure 4.9(b), whereas the result of the exploration algorithm are presented in figure 4.9(c). From figure 4.9(c), two frontier regions were detected at this stage, which are represented by green colour. Frontier centroids were found for each region. The prospects were measured for each region. Two prospects values are applied in this situation, navigation cost and feature visibility around each region. The robot found that the first region had a total prospect value of $E_1^{dl} = 0.5$, whereas the second region had a total prospect value of $E_2^{dl} = 0.1$. As a result, the robot chooses the best prospect to continue achieving the task. The robot plans a path to this target, and then navigates to it as the second place of the exploration.

(a) (b) (c)

Figure 4.9: The results of the exploration algorithm at the first position: (a) the position of the robot, (b) the result of the extraction algorithm, (c) the result of the updated map

Figure 4.10 shows the result of the second step at the second position of the exploration. In this stage, the robot extracts features from the back view and front view as shown in

figure 4.10(b) and 4.10(d). Repeating the same procedure of the algorithm in figure 4.6, two frontier regions are found at the evaluation stage. The first frontier region has a total prospect value of $E_1^{dl} = 0.4$, whereas the second frontier region has a total prospect value of $E_2^{dl} = 0.2$. As a result, the robot moves straight forward once more through the corridor to the third position.



(a) (b) (c)

(d)

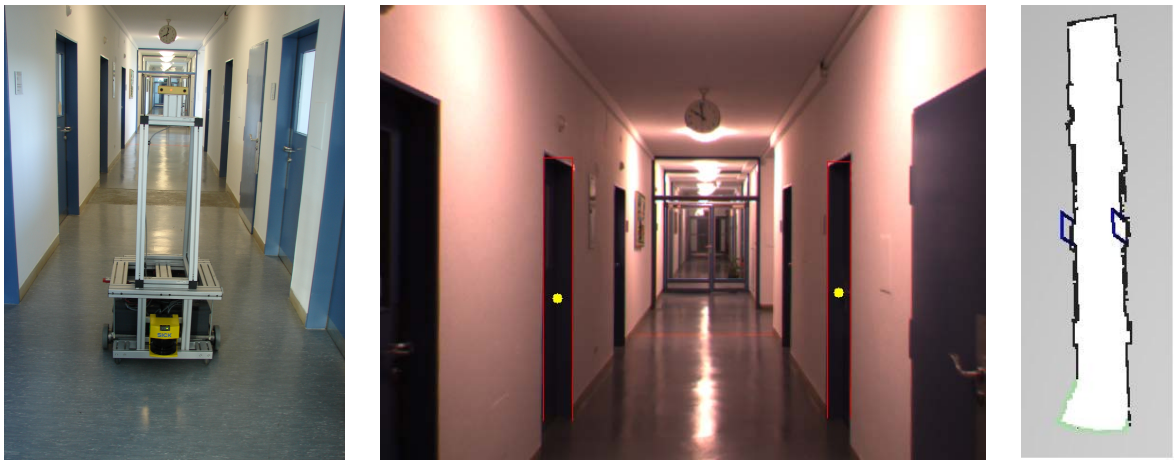
Figure 4.10: The results of the exploration algorithm at the second position: (a) the position of the robot, (b) the result of the extraction algorithm (front view), (c) the result of the updated map, (d) the result of the extraction algorithm (back view)

Figures 4.11 and 4.12 show the result at the third and the fourth position of the exploration task, respectively. As a result of step 3, only one frontier region is detected as shown in figure 4.11(c), thus the robot navigates to this target to decide if there is any other region to be explored. The result of step 4 showed that there are no more regions to be explored. Thus, at this place the exploration strategy algorithm is finished and the robot is going into idle state. Figure 4.12(c) shows the global map of the chosen environment, containing all the doors extracted during the execution of the task.

As a general result of the general approach which uses two types of the prospect values, it could be shown that the robot can explore the rest of the environment efficiently. With the experiment, ten doors and one fire-extinguisher were extracted correctly from different views

of the robot position as presented in the figures 4.9(b), 4.10(b), 4.11(b) and 4.12(b). Only one door was not extracted due to the occlusion of this feature.

The features extracted during the exploration algorithm are used in the SLAM algorithm to predict and update the position estimate of the robot at each time step. Each feature is represented by a point feature by taking the corner detector of the upper side of the feature itself. If the observation of this feature is new, then an initialization process takes place by using equation A.25. Otherwise, the measurement observation is used to update the filter and to correct the estimate of the vehicle.



(a) (b) (c)

Figure 4.11: The results of the exploration algorithm at the third position: (a) the position of the robot, (b) the result of the extraction algorithm (back view), (c) the result of the updated map

(a) (b) (c)

Figure 4.12: The results of the exploration algorithm at the fourth position: (a) the position of the robot, (b) the result of the extraction algorithm (back view), (c) the result of the global map

4.4.2 Result of the Master Approach

The experiments were conducted in an environment of the size $20 \times 20m$, with a robot speed of $0.2m/s$. Two robots with the same specification were used to present the idea of the algorithm shown in figure 4.6. Besides the navigation cost and the feature visibility, the third prospect value of coordination was also used in this situation. Such a prospect lets the robots distribute themselves over the environment efficiently avoiding any work conflicts or sensor interference problem.

The task of the exploration is achieved by two robots in three steps. At each step, each robot applies the sequence demonstrated in figure 4.6 and 4.7. In the step of the evaluation of a region, each of them must consider if the region was selected as a target by the other robot or not.

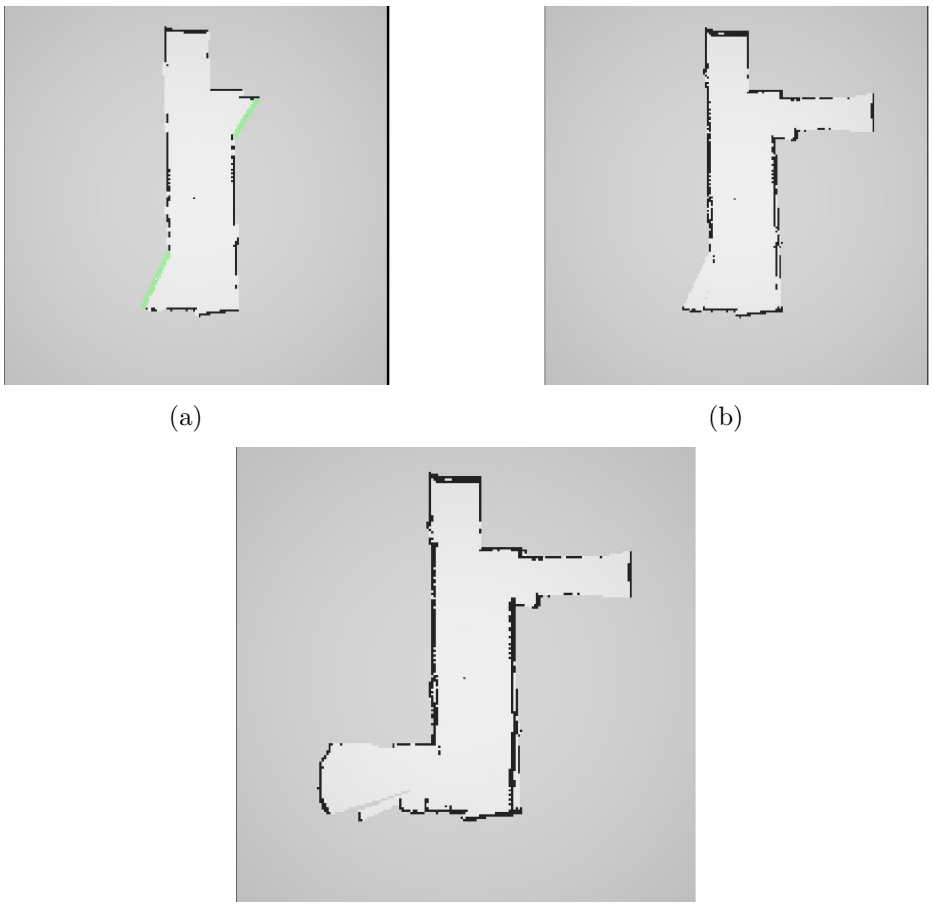
Figure 4.13 shows the exploration result achieved by each of them. The first master starts its sensor sweeping at the first position, it finds two frontier regions to be explored as shown in figure 4.13(a). It applies the prospect values, chooses one of the frontier regions as the next destination and navigates to it. The second master receives an acknowledge that the first region is already selected by the first master. Thus, the second master deletes this area from the list of potential targets, measures the prospect values of other regions, and navigates to the best one as next target. Figure 4.13(b) shows the result of the second master.

Figure 4.13(c) shows the global map composed by the first master. In this stage, two tasks are performed by the first master. Firstly, it explores the first area detected by it at the first stage. Secondly, it fuses the local map generated by it and the local map explored by the second master into a global map.

Figure 4.13(c)shows the final global map of the environment, with no more areas to be explored. As a general result, the time of the exploration is shortened because of the execution of two different tasks at the same time, and this would in particular be an important factor in a large environment. One more important point of doing this experiment is that the robots distribute them selves over the environment efficiently because they use the three prospect values explained in section 4.2, i.e. the use of the suggested prospects accomplishes the desired results, namely to direct the robots into places which need to be explored without any conflicts while performing the task.

4.4.3 Result of the Slave Approach

The goal of the exploration algorithm is to send the slaves equipped with a moving 2D scanner, the RoSi system as presented in section 6.5 to some places in the environment, in order to get a 3D map of these destinations. Figure 4.14 shows a sample of range images with 8 meter maximum range and the corresponding original image captured with a digital camera. Figure 4.14(b) shows the result obtained including the texture information of the surroundings, whereas figure 4.14(c) shows the same case but without introducing any extra information. Different views of both situation are also shown. The process of building a surface model and the fusion of different data of different places acquired by different slaves are presented in chapter 6.



(a) (b)

(c)

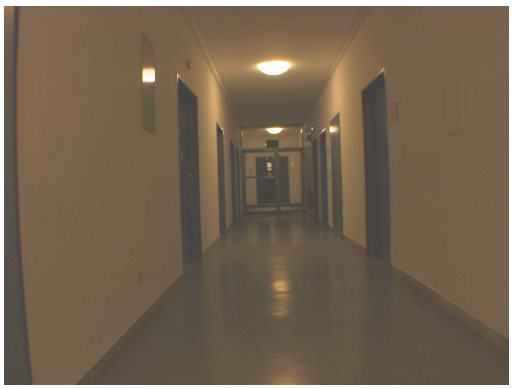
Figure 4.13: The results of the exploration algorithm: (a) the result of the exploration algorithm achieved by the first master, (b) the result of the exploration algorithm achieved by the second master, (c) the result of the global map composed by the first master

4.5 Conclusion

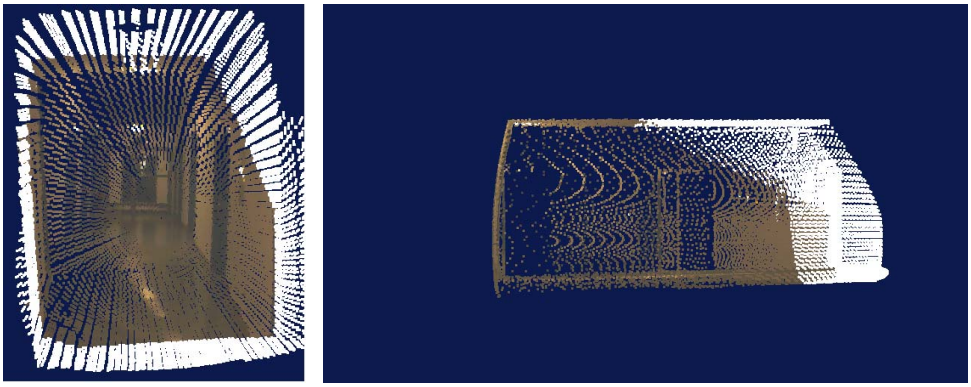
This chapter presented the strategy of the exploration algorithm using multi-mobile robots. It started by introducing the general concept of how to acquire a 3D world model from various sensors.

The second section introduced some prospect value functions used in our strategy, showing their theoretical aspects and how they are applied. Within this section, the idea of the frontier regions was presented, and different prospect values were defined, in order to compute the evaluation degree for each region to be selected as a target of the next move. The navigation cost, the feature visibility around the area to be explored and the coordination factor are three prospects, which are used to calculate the total prospect of each target in order to choose the best one of several interest regions.

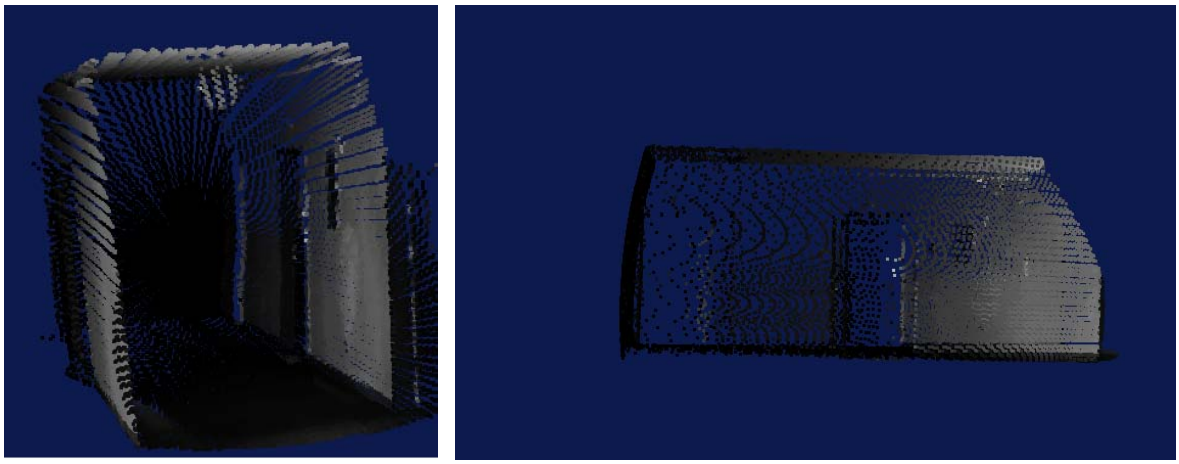
In the third section, we presented the exploration algorithm for three different cases. Firstly, the exploration algorithm of a single robot was demonstrated to explore the environment efficiently based on the prospect functions presented in section 4.2. The second case used two robots as masters. The appropriate algorithm was presented, showing how the prospect



(a)



(b)



(c)

Figure 4.14: The result of the exploration algorithm (slave case): (a) the target position of the slave, (b) the range data with texture overlay, (c) the range data without texture overlay

functions are applied in this case. Thirdly, the exploration algorithm in the case of the slaves was also demonstrated.

In the fourth section, the experimental results obtained using Odete2 platforms were presented. Two different cases were presented (one robot and two robots as master). This section defined how the prospect values are used in each case, and how they are useful in order to achieve the task of the exploration efficiently. The task done by slaves was also presented within this section.

Chapter 5

Cooperative Strategies

5.1 Cooperative Robotics and Communication

Cooperation among mobile robots is becoming more necessary as tasks increase in their complexity. One single robot can hardly perform every task just as humans specialize on different abilities e.g. when consulting a large building. As technology advances in the field of computing and wireless communication, cooperative robots are becoming more of a possibility than ever before. Complex tasks one robot is not able to perform on its own can be carried out through close cooperation of multiple robots. However, cooperating robots are not easy to control as a single cohesive unit. The robots must reliably communicate with each other. They must know what the other is doing, where it is, and where it is going to. All of this information must be used to solve the coordination problem in order to perform a task more efficiently.

Cooperative robots must take care to avoid collisions with their team-mate. That means, they must perform their planning in such a way that it does not conflict with other robot's planning in order to reach their destination safely. They must also take care to avoid performing tasks which were already completed by other robots.

Fully autonomous teams of robots could accomplish complex tasks that one robot is unable to achieve, tasks that are too dangerous for humans, or they could simply increase the productivity and the efficiency of large projects. To make a team of robots cooperate to achieve such tasks, a framework and its software structure must first be build to allow for a reliable communication between the robots in order to allow them to achieve the tasks in a robust manner of cooperation.

Within our work, the architecture framework was developed with built-in multi-robot capabilities built in. The architecture is based on a master-slave approach, which allows the robots to accomplish a tightly coupled exploration of the environment. The communication module within this structure allow for robot-to-robot communication using different types of protocols in order to let the robots cooperate in an efficient manner. The framework also deals with the problem of the coordination of robots to avoid collisions in situations where two robots are performing different tasks close to each other.

As the task of our work is the exploration, the mechanism of coordination developed within this framework allows the robots to explore different regions in such a manner that each one of them explores a region yet unexplored by any other robot. The key in the solution of this planning problem is provided by the help of the master, which directs its team to choose an appropriate area for each slave, such that it does not explore a region that has already been covered. Beside the master assistance, the slaves follow the coordination technique explained in chapter 4 in order to distribute themselves efficiently over the environment.

To design a cooperative multiple robot system, it is desirable to find a description taxonomy for describing and classifying the multiple robot system along the main axes of the research in exploration. Farinelli et al. [Farinelli 04] presented a taxonomy of multiple robot systems focused on coordination mechanisms, using a top-down approach to refine the level of the system's structure representation. Figure 5.1 presents their taxonomy, which includes four different representation levels: cooperation level, knowledge level, coordination level and organization level.

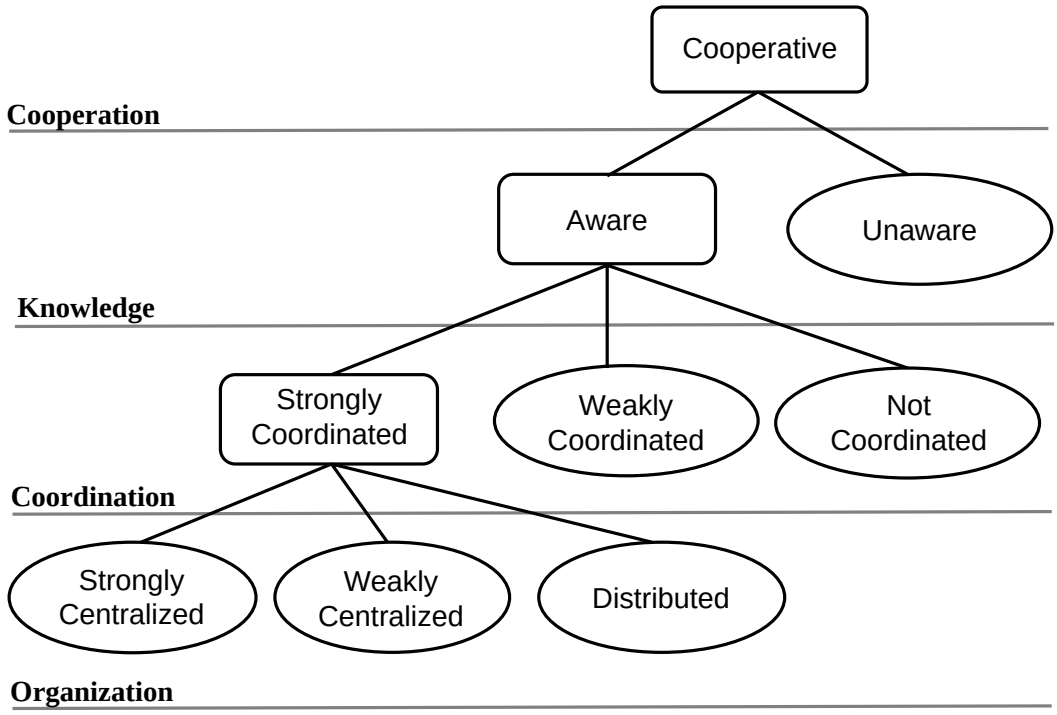


Figure 5.1: Classification of multiple robot systems focused on coordination. Figure reproduced from [Farinelli 04].

The *cooperation level* is a situation in which several robots operate together to perform some global task that either cannot be achieved by a single robot, or for which execution can be improved by using more than one robot, thus obtaining a higher performance.

The *knowledge level* characterizes how much knowledge each robot has about the presence of other robots in the environment, i.e. its awareness of them. Completely unaware robots act without any knowledge of the other robot in the same environment. Cooperation among unaware robots is classified as the weakest form of cooperation.

The *coordination level* is concerned with the mechanisms used for cooperation, in which

the actions performed by each robot take into account the actions executed by the other team-mates, in such a way that the group operates in a coherent and efficient manner.

The mechanisms to achieve coordination between the robots can vary. The simplest case is an aware, but not coordinated approach which allows only simple tasks to be carried out by the robots. This type of cooperation reduces the communication failures which may occur. However, the robots need sometimes at least to be weakly coordinated in order to avoid sensor interferences. An even more abstract strategy are aware, strongly coordinated robots. Within this strategy, a robust cooperation can be achieved, in which the robots distribute themselves over the environment in order to complete the global task efficiently.

The last level in this taxonomy is *the organization level*, which characterizes the way the decision system is organized, i.e. whether it is centralized or decentralized. In strongly centralized systems, a master-slave approach is used. The master is in charge for organizing the work of the entire team, while the other members act according to its directions. On the other hand, a decentralized system is composed of several slaves which are completely autonomous in the decision process with respect to each other; in this class of system, only a list of tasks to be carried out is sent from a master to its slaves.

Farinelli et al. [Farinelli 04] mention one more taxonomic description, which plays an important role to the previously presented taxonomy, namely *communication*. The type of communication has a strong influence on the coordination and organization levels. For example, strongly coordinated systems necessarily require an extensive communication. Without any communication mechanism, the cooperation among robots is difficult to perform. By exchanging data among the robots in a team it is thus possible to cooperate efficiently. In multi-robot systems, three general approaches have been used:

- No communication
- Passive communication - sensing
- Active communication

Using no communication at all the multi-robot system is a collection of single robot systems. For some tasks such as the box-pushing problem this might be satisfactory. In systems with passive communication the robots simply sense either the actions of the other robots or the positions of their team-mates. In both situations, it is sometimes necessary to have some assistance tools in order to achieve the sensing task. The last approach is an active communication between the robots through a network. For an exploration task with a high level of coordination, the active communication is the most appropriate. To achieve this task, the robots need to known what areas have been covered so far and where to proceed. This information is exchanged among the robots more efficiently with a means of active communication.

Within our work, passive and active forms of communication are used to achieve the cooperation task. With the passive communication, the master sets its position at the starting time as a reference point for the multi-robot system. It uses its stereo camera sensor to detect the positions of its team members based on the coloring of its slaves. The slaves themselves

can also estimate their position by using their RoSi system. Using the remission factor and the beacon landmark, the slaves calculate their position. On the other hand, in the case of active communication, the respective measurements are exchanged between master and slave.

5.2 Active Communication

5.2.1 Analysis of the Cooperation among the Robots

Active communication is a design issue for the cooperation of multi-robot systems. The communication is taking place directly via an explicit communication facility, namely *a wireless network*. To perform cooperation, protocols are required which allow the robots to operate without any human control. To design these protocols, we need to understand the cooperation scheme among the robots.

Figures 4.6, 4.7 and 4.8 present exploration strategies of several robots using a master-slave approach. Within this work, three scenarios are used in order to analyze the cooperation process among the robots, and to define the type of information to be exchanged between the robots. The scenarios are master-to-master communication, master-to-slave communication and slave-to-master communication.

Figure 5.2 presents the description of the control sequence of a master robot only with regard to the communication with another master in a cooperation scheme between them. A complete task is illustrated in figure 4.6 and 4.7.

A system of master robots executing this type of algorithm requires the following skills of each robot:

- *Receive acknowledgement:* The form of the received data is the position of the sender's next move. The received robot acknowledges that this position is chosen as the next target by the sender. Thus, the received robot will take this acknowledgement into account when it decides to select its next destinations.

- *Send Acknowledgement:* The form of this data is the position after the next move of the sender, which gives an indication to the receiving robot that this position is selected by the sender to be its next target. Thus, the other robot cannot select this position or any other position in the area around it that is covered by its sensor.

- *Receive edge map:* The received data is the local 2D map of the sender. The map is represented in the form of edges to save bandwidth cost of communication. When this map is received, the robot needs to perform a simple reconstruction of the occupancy map based on these edges as well as on the scanning position from which the edges were determined. So, it fuses this map into the global map it has captured by itself. This map is necessary to know if any place in the environment needs to be explored or not.

- *Send edge map:* The data is represented as a local 2D map of the place where the robot is currently standing. The map is sent to the other robot to be added to its global map, and to allow the receiver to conclude if any other area needs to be explored.

- *Receive scanning position:* The form of received data is the position of the scanning done by the sender. It is needed in order to fuse the received 2D edge map into the global map.

- *Send position of scanning:* The position of the robot when performing the scan is transmitted. This is to be used in combination with the edge map in order to let the other robot know the structure of the actual local map already built by the sender.

Master-to-Master program:

1. Listen for robot acknowledgement
2. On robot acknowledgement,
   a. Receive acknowledgement
   b. Receive edge map
   c. Receive position of scanning
3. Check task available
4. On task,
   a. Send acknowledgement
   b. Send edge map
   c. Send position of scanning
5. Perform action
6. Update
7. Go to 1

Figure 5.2: Description of the data exchange between the master robot systems focused only on the cooperation scheme between them

Figure 5.3 presents the second and third scenario, which illustrates the description of the control sequence of the communication between the master and the slave, focusing only on the cooperation scheme between them. A complete task is illustrated in figure 4.6, 4.7 and 4.8.

A system composed of a master and a slave robot executing this algorithm requires the following skills of each robot:

- *Send selected task to slave:* The form of this data is the position of the next move to be performed by the slave. This command is sent by the master in order to direct one of its team members to a certain place in the environment to build a 3D map of this region.

- *Receive command:* The received data is the target position of the next move as a command order from the master to let this slave move to this destination.

- *Send (Receive) FB:* The form of this data is a feature-based map, which is sent by the master and is received by one of its team members to be used in the localization process.
- *Send (Receive) OG:* The form of this data is an occupancy grid map, which is sent by the master and is received by a specific member of its team. This map is to be used by slave for the path planning process.
- *Send (Receive) point clouds:* The data transmitted is a range image in the form of point clouds. This data is sent by slaves and is received by their master, after it took a 3D map of the target region the master performs the merging process of different range images received from different slaves at different positions.
- *Send (Receive) a scanning position:* The position of the slave at the time of scanning is needed by the master in order to finish the merging process mentioned in the last skill.

Master-to-Slave Program:	**Slave-to-Master Program:**
1. Check for task available 2. On task, a. Select a task to be sent to slave b. Send selected task to slave c. Send FB (feature-based map) d. Send OG (grid-based map) 3. Check for result available from slave 4. On result, a. Receive point clouds b. Receive position of scanning 5. Go to 1	1. Listen for command received 2. On command, a. Receive command b. Receive FB (feature-based map) c. Receive OG (grid-based map) 3. Perform action 4. Check the finishing action 5. On finish, a. Send point clouds b. Send position of scanning Go to 1

Figure 5.3: Description of the data exchange between the master and slave robots focused only on the cooperation scheme between them

As a general conclusion of this analysis, the types of data to be exchanged among the robots is limited to two main types. Firstly, there is data in the form of a position, which can be represented either as the actual position of the robots, as a command to any slave, as an acknowledgement from one robot to its team that this position is the target of the next move, or as a scanning position to be used to fuse different type of maps into a global map. Secondly, there is data in different forms of maps, which can be exchanged among the robots as files. The map can be represented either as an edge local map, as an occupancy grid map, as a feature-based map, or in range image form.

Thus, we need to design two different protocols. The first protocol should concern with the transmission of positions, whereas the second protocol should take care of the exchange of different types of maps among the robots. In order to let the robots differentiate among such different data of the same category, we introduce an ID number to each process to identify the specific representation type of the data. Tables 5.1 and 5.2 summarize the different representation types and their respective ID numbers.

Map ID	Map Type
0	Edge map
1	Occupancy grid map
2	Feature-based map
3	Range Image point clouds

Table 5.1: Parameters of the map types and their IDs

Position ID	Which Position?	Comments
0	Actual Position of sender	
1	Destination of the master (sender)	use it as acknowledgement command
2	Destination of the slave	use it as direction command to slave
3	Scanning position	to be used with the edge maps
		or range images

Table 5.2: Parameters of the position type and its ID

5.2.2 The Communication Protocol

Client-Server Model

A common model of communication is client-server. This model is widely used in a number of network applications such as the world wide internet. One of the main reasons for choosing client-server approach is that it is easy to create and maintain. Another practical reason is that our modular architecture software (MCA) uses a client-server model as well, with all communication being performed over TCP (Transmission Control Protocol) sockets and binding for a variety of debugging process or interpart connections. TCP/IP establishes connection between server and client; setups input and output stream and sends/receives data.

In the MCA architecture, a debugger tool is used, the so-called MCAbrowser. It connects via TCP/IP to processes, visualizes the hierarchical control system and provides access to the module interface. It enables fast and efficient program analysis, debugging and test.

Furthermore, the client-server model is used within the MCA architecture to realize data transfer. This is done by using interpart connections. The architecture combines parts that are developed separately like e.g. the vision system and the basic motion control strategy system of a mobile robot.

To realize our master-slave approach with the client-server model, we built up one server running on the master, which is able to connect to several clients. These clients represent the master and all slaves of its team. That means, the central server running on master receives both transmission requests from the master (represented by one client) and transmission

requests from slaves (represented by another client), and it services these demands in order to send the data to its destination. Thus, a central server maintains the information about the master and its team of slaves in order to administrate any requests.

The term server employed in the client-server model is sometimes confusing with other structures of the central demand application, so we refrain from using this term in order to avoid any confusion as to the source of the information that each client receives. We then use the term responder to denote to any request it receives.

Messaging Protocols

A protocol defines the way in which nodes can communicate over the network, e.g. in a cooperation between master and slave robots. In order to make communication easy and efficient, it is important to design a simple protocol that can exchange different types of information among the robots without any complexity in the message packages.

A result of the considerations in section 5.2.1 is that only two types of data need to be exchanged among the robots, namely positions and maps. Thus, different types of messages are built in order to achieve these tasks. The classification of these messages is performed based on their functionality. Table 5.3 lists all types of messages with their descriptor and description.

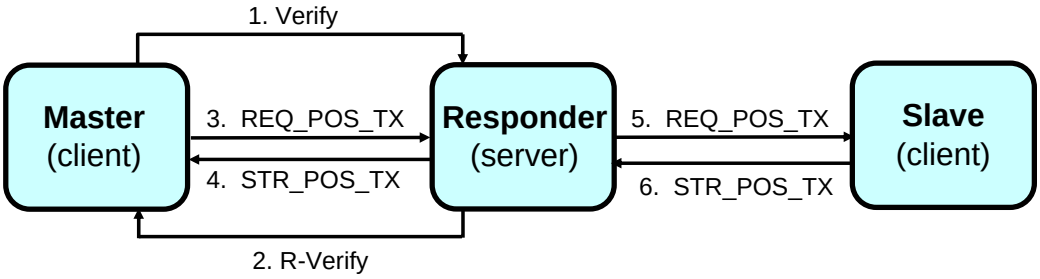
No.	Descriptor	Description
1.	REQ_POS_TX	request for position transmission
2.	REQ_MAP_TX	request for map transmission
3.	STR_POS_TX	start servicing the position request transmission
4.	STR_MAP_TX	start servicing the map request transmission
5.	COMP_MAP_TX	an acknowledgement that the map transmission is complete
6.	Acknowledge	an acknowledgement that the data is sent
7.	Verify	verify availability of a particular robot (destination)
8.	R_Verify	response to verify

Table 5.3: Different types of messages

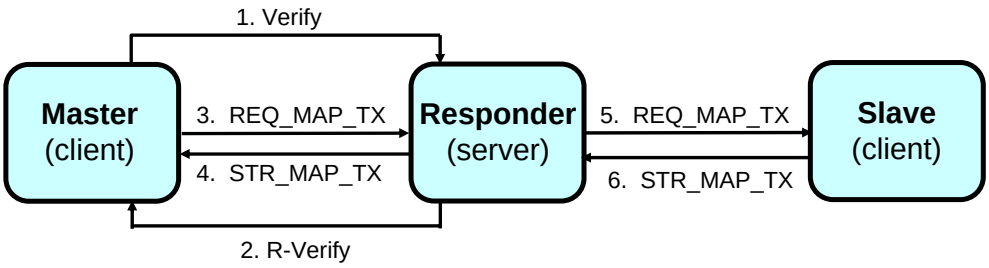
Figure 5.4 shows the communication between master, slave and their responder using the different kinds of messages presented in table 5.3. In order to send a position or a map, a sequence of messages is exchanged between the master, the slave and their responder. As shown in figure 5.4(a), a position is sent from master to slave. The sequence starts with the master verifying if the destination client is connected (No. 1) and the responder acknowledges this message and sends a reply (No. 2). Then, the master requests for position transmission (No. 3) and gets the reply (No. 4) that it can start the transmission. Once the responder gets an affirmation from the master that the data is sent, the responder starts to service this information to be sent to the destination. It passes the request for position

transmission (No. 5), and the slave acknowledges this message and sends a reply (No. 6). Once the slave gets its data, it sends an acknowledgement to the responder that the data is received correctly.

Similarly, figure 5.4(b) shows the communication between the master, the slave and their responder in the case of a map transmission. The transmission in the reverse direction from slave to master is done in the same manner following the same procedure and messages protocol.



(a)



(b)

Figure 5.4: Illustration of different types of messages: (a) in the case of a position transmission from master to slave and (b) in the case of a map transmission

The transmission operation from master to responder and from responder to slave is asynchronous. That means, the responder saves all the received requests and the corresponding data into a queue, and it starts sending them to their destination in sequences in the form of a first in - first out stack. If there is temporarily no reply from any one of the target destination, it reschedules this request as the newest request it receives and continues with the other request demands.

It was shown above (cf. table 5.1 and 5.2) that we need an ID type of the data to define the kind of position or map to be transmitted. Also necessary is of course the destination address in order to achieve the request successfully. Thus, every message has a descriptor header that contains some pre-defined information. This header determines the ID type of the data to be sent and the destination address (cf. table 5.4).

ID type	Destination Address	Data to be sent

Table 5.4: The format of the data message

Figure 5.5 shows a typical example of message sequences with a master requesting for a position or map transmission to the responder. The master first sends a message to test the availability of its destination. Once the responder verifies it, the master can then send a request to transmit either a position or a map.

Figure 5.5(a) presents how the position is transmitted, also showing the format of the data message, whereas figure 5.5(b) shows the map transmission sequence. The transmission from the responder to the slave then has the same structure as the master to responder sequence.

In fact, figure 5.5 shows the time state diagram that is required to send information between different parts. It is clear from figure 5.5(b) that a large time is needed in order to send a complete map. Thus, in order to use the communication bandwidth as efficient as possible, edge maps are used in order to perform the communication process more efficiently.

The protocol was implemented under the environment of MCA and was tested using two masters in order to explore a part of the environment. It showed that the protocol is working well without any delay or any problem of packet loss. The result of figure 4.13 was achieved by applying this communication protocol which used all the messages stated in table 5.3.

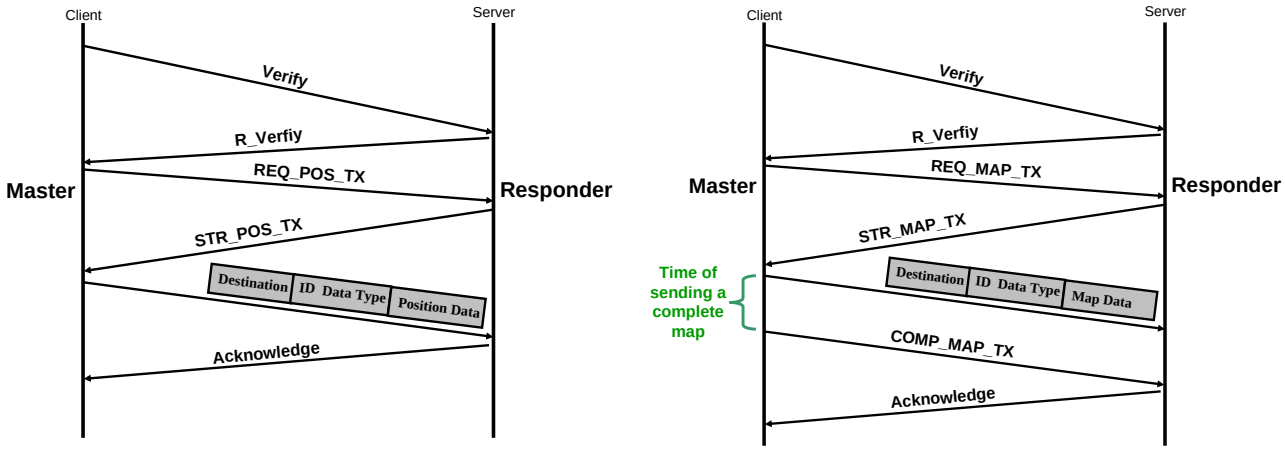


Figure 5.5: Typical message from master to responder: (a) in the case of a position transmission and (b) in the case of a map transmission

5.3 Passive Communication

Another way to perform a cooperation as discussed in section 5.1 is by using a passive communication. In communication literature, passive communication is also known as implicit communication, which is the transmission of information through the robot's sensors. For example, Rekleitis [Rekleitis 00] used pairs of robots that observe each other's behaviour, complementing each other to reduce odometry errors. Each robot was equipped with a robot tracker sensor that observed the other robot and reported its relative pose. The observing robot used the position of its partner in order to update its own position.

An interesting fact about passive communication is that no failure in transmission is to be taken into account, and it does not need any extra hardware such as a wireless network

card in order to interchange some information. Furthermore, the sensor used in the passive communication can at the same time also be used in other tasks such as exploration. For these reasons, passive communication is extremely robust.

However, there are some limitations to the use of passive communication. Firstly, the communication is limited by what a robot can perceive using its own sensor. Secondly, it needs some setup equipments in order to receive any information from the environment. Thirdly, its robustness depends more on the type of task would be achieved.

Different types of sensors can be used to perform a passive communication, in particular cameras and laser scanners. Rekleitis [Sim 01] used a tracking device that can estimate the position and orientation of a mobile robot relative to a base robot equipped with a laser range finder. He used two robots, where one is moving and the other one is stationary. The second one acts as an artificial landmark in order to be detected by the moving robot to recover its pose with respect to the stationary one. He used a three plane target mounted on the observed robot in order to estimate its pose (see figure 5.6).

Figure 5.6: Robot tracker: The observed robot with white target on the left and the observing robot with the laser range finder on the right. Presented in [Sim 01].

Rocha [Rocha 05] used two robots in order to perform a mapping task. He localized his robots by using a fixed external camera and color objects mounted on the robots. Within his work, the data captured by the camera is processed by an external computer. The result is a pose estimation of these robots with each map update.

Jung and his group [Jung 97] used a vision system in order to perform a cooperation task between two robots. Within this work, each robot observes its partner by matching a unique geometric pattern mounted on the sides of each robot. To cooperate, one robot observes the other robot's actions and can then determine the approximate location of the litter deposited by the first one. So, the cooperation is supposed to improve the efficiency of the cleaning task.

There are three benefits for robots to use a passive communication. Firstly, the odometry measurement can be corrected by providing the measurements relative to the position of a

stationary robot. Secondly, by using a team of robots it is required at the beginning to know their initial position with respect to one of them as reference coordinate. Thirdly, when a robot can not determine its position correctly any more, it can use passive communication to predict the position of one of its team members in order to estimate its own position.

Within our work, we have tested the second and third benefits using different types of sensors. The master is equipped with a stereo camera. It can detect the position of any slave by using colored objects placed on the robots. The slave is equipped with a RoSi system. By using this system and some artificial beacons placed on the master, it can detect the position and orientation of the master. The detection process is based on a particular property of the laser scanner, the remission factor. In the next subsection we present both detectors in more detail.

5.3.1 Visual Detector

A variety of sensing technologies could be used as a robot detector. Our implementation of the robot detector sensor is based on the visual detection of a color cube target pattern mounted on the slave robot. The master robot is equipped with a stereo camera that allows it to detect its team very accurately. The slave is marked with a special pattern of three cubes with known color. Two of them are placed directly above the axle of the robots. They can be used to detect the position of the robot correctly. The third cube is placed in the head direction of the robot in order to define its orientation of the robot. All three cubes together form a triangle

different designs for these cube objects are possible, but they should satisfy two key requirements: The cube objects should be robustly detectable and they should be helpful to easily estimate the position and the orientation of the robot on which they are mounted. Figure 5.8 shows the color object target mounted on a slave in a certain configuration.

(a) (b) (c) (d) (e)

Figure 5.7: Different situation of possible values of robot orientation: (a) no rotation at the beginning, whereas in all other cases the robot is rotated by (b) 45°, (c) −45°, (d) 135° and (e) −135°.

The used target consists of three color cube objects distributed in a triangle form on a slave. These objects are always visible from any point around the robot. Based on this configuration, the slave position can easily be calculated. It is equal to the position of central point of the base axle of the robot. Different forms of rotation are visualized in figure 5.7 to be taken into consideration when the orientation is estimated.

Algorithm 5.1 presents the sequence of how a robot can be detected by using a stereo camera and a target composed of three color cubes. The input to this algorithm is the position of these three objects detected by a stereo camera based on their color and geometry. The algorithm's result is the position and orientation of the slave. If one of these objects is not detected by a stereo camera, then another frame is captured and the algorithm is processed once more. The position value is determined by finding which two of these objects are forming the triangle base, then the origin of the local coordinate system is the center of this base line. On the other hand, the orientation value is calculated depending on the direction of the detected robot and the orientation of the detector as well.

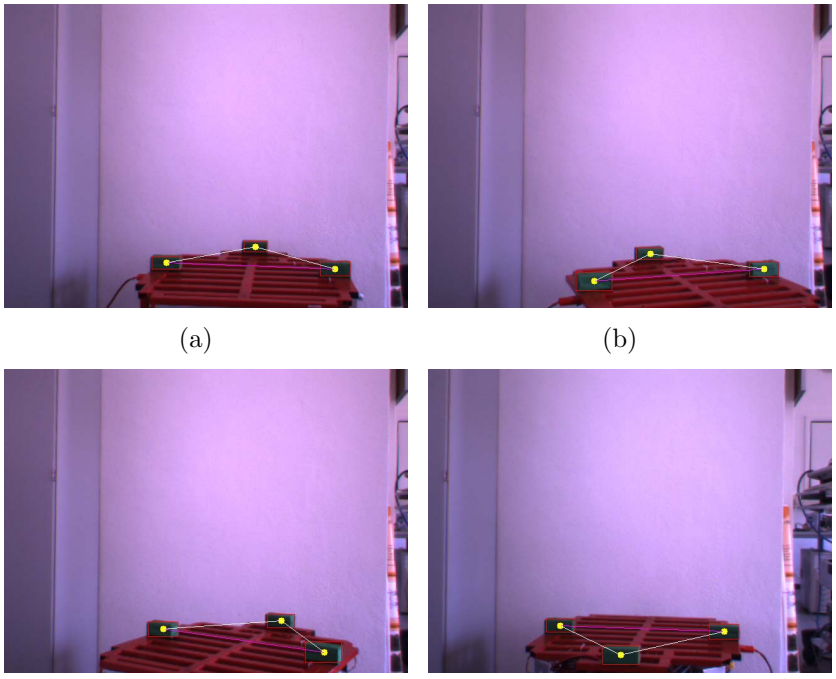
Input: Position of artificial objects
/*3 objects represent the head and position of the robot*/
Output: Pose(x, y, θ) /*The position and orientation of the robot*/
1: **if** *not InitializedState* **then**
2: InitializeState( )
/*the detector builds an initial state as a reference based on its first position and orientation*/
3: **end if**
4: **if** *all objects on a robot are detected* **then**
5: Find the axle base
6: Determine left and right side and header point
7: $x \leftarrow (Leftpointofaxlebase(x) + rightpointofaxlebase(x))/2$
8: $y \leftarrow (Leftpointofaxlebase(y) + rightpointofaxlebase(y))/2$
/*Get the x and y position of the robot based on the axle base of the robot*/
9: Calculate the slope m /*This is used to calculate the orientation*/
10: **if** *Header point is directed to the same direction as initialized state* **then**
11: $\theta = \arctan(m)$ /*Situation amends to $+45°or-45°$ */
12: **else**
13: /*header point is directed to opposite direction as initialized state*/
14: **if** $m < 0$ **then**
15: $\theta = PI + \arctan(m)$ /*Situation amends to $+135°$ */
16: **else**
17: $\theta = -PI + \arctan(m)$ /*Situation amends to $-135°$ */
18: **end if**
19: **end if**
20: **end if**
21: **Return** $Pose(x, y, \theta)$

Algorithm 5.1: The outlet algorithm of the position and orientation detection of the slave robot

Different experiments were performed in different situations. Figure 5.8 shows some of these results presenting four different situations. In each time the position and orientation are detected correctly with an accuracy of $5mm$ in position, and with $2°$ in orientation.

However, some disadvantages need to be taken into account. Firstly, the target can not be detected from a large distance away from the master, or from a very small distance near the master. From experiments, we have found that the potential range falls between $2.5m$ to

$6.5m$. Within this range, the master can detect the slave easily. The second one is related to the stereo camera itself. To calculate a 3D correspondence using this camera, the objects need to appear in both left and right images. Thus, a master must choose a good position in order to detect the slave correctly.



(a) (b)

(c) (d)

Figure 5.8: Detection of the robot orientation based on visual targets: (a) the orientation is 0°, (b) the orientation is 25°, (c) the orientation is −20°, (d) the orientation is 175°.

5.3.2 Laser Detector

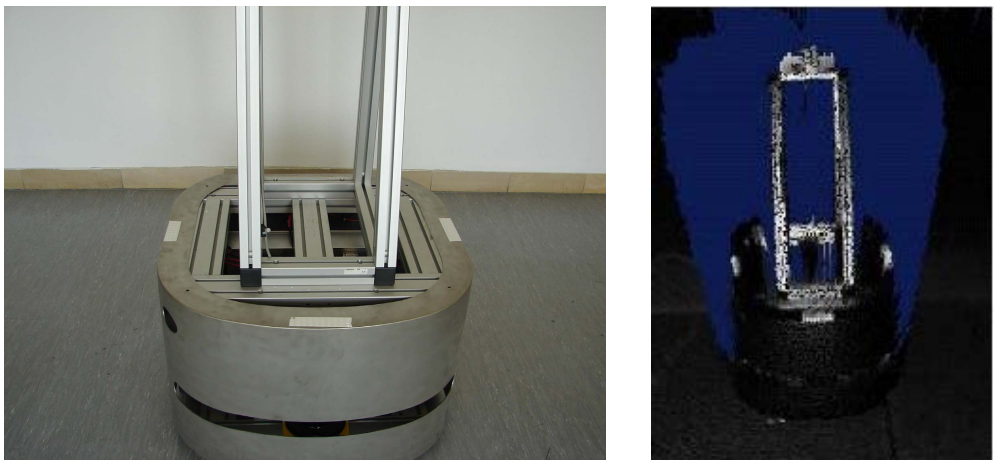
The second implementation of the robot detector sensor was based on the RoSi system and a target mounted on the detected robot. The robot detector is implemented by using a laser scanner of the RoSi system mounted on the slave and three artificial beacons mounted on the detected robot (see figure 5.9(a)). This type of target was used in order to permit an accurate and robust estimation of the position and orientation using a laser scanner.

Laser scanners are active sensors, that emit light and therefore do not require specific lighting conditions. By analysing the remission values, i.e. the amount of light that is reflected by an object, it is possible to also get information about the character of the object's surfaces.

The amount of light reflected (i.e. the remission factor) by an object is very dependent upon the texture and the smoothness of the surface. We have found that, the artificial beacon we have used, has a very high remission value compared to other objects found in the environment. Thus, detecting such beacons is relatively easy. Figure 5.9(b) presents the range image data of the master taking into consideration the remission factor. The three beacons are robustly detected in these images.

To reuse the algorithm developed for the case of visual detector, which correctly detected the position and orientation of the target robot, these beacons are distributed on the robot in a

form of a triangle, where two of them represent the axle base and the third one represents the head of the robot in order to estimate its orientation.



(a) (b)

Figure 5.9: Detection of position and orientation of a robot based on a remission factor: (a) the artificial beacons placed on the master, (b) the result of scanning - the position of the artificial beacons is detected.

5.4 Cooperative Strategies – Master Case

In chapter 4, we presented how the robots can perform the exploration task based on an evaluation function composed of three prospect values, namely navigation cost, the visibility of the features, and the cost of coordination. These prospects have been tested in experimentally with two robots and a limited size of the environment. To study the efficiency of these prospects, we have implemented a simulation package to study in more details the performance of the system applying such prospect values.

Three strategies are used in order to evaluate the performance of the system in the master case. The first strategy performs the exploration using the frontier mechanism and prospect values mentioned in chapter 4, which leads the robots to explore different regions simultaneously. The second strategy extends the first one while trying to avoid its limitation. In this strategy, the exploration task is done by assigning only a limited area to each robot for exploration. In the third strategy, a dynamical technique is used in order to let the robots cooperate more efficiently, but still finish the task at the same time. To evaluate these strategies, we have modelled two different office and laboratory buildings to perform the simulation.

5.4.1 Frontier Strategy

The frontier strategy results from the idea of finding a frontier area as presented in chapter 4. If a master detects two frontiers in two different directions, it then applies the prospect function on these two regions and chooses the one who has the best prospect value to be

explored in the next move. It sends a message to the second master to inform it which area it chose. The second master can now calculate the prospect values of the remaining region, and explore the region with the best score.

So, the idea of this strategy is to split the area into two sub-areas, and both robots start to explore their respective sub-area to find any interesting target area to be explored by their teams. I.e. if this strategy is performed by two robots, both of them explore different sub-areas simultaneously.

In each step of the exploration, each robot has to perform the following tasks:

- Check the presence of natural landmarks and locate them.
- Exchange messages with each other, in order to bring the available information about the environment up to date.
- Check if there is another area to be explored.
- Move to a new area if there are more places to be explored. Chooses the next move based on the prospect function of these places.

Figure 5.10 illustrates the frontier strategy in the case of splitting the area in two parts. The robots have to navigate through the corridor of an unknown floor of a building and search for possible targets such as doors to be used as landmark.

In order to find out the exploration time needed to achieve the task, we assumed two conditions. The first condition is that the exploration time is calculated without considering the time required to do a task, where as the second condition is assumed to contain the time of doing a task.

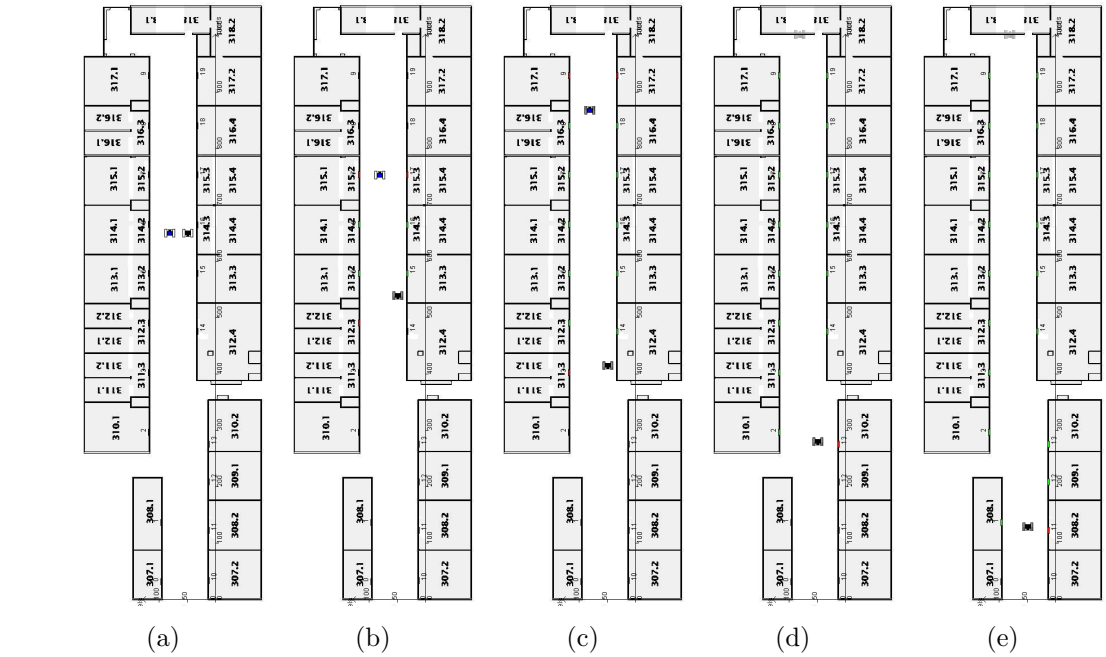
The total time of exploring the whole area depends directly on both times required by the first and the second master (A and B) to do a movement step t_A and t_B respectively. Knowing the speed of the robots, we can estimate the times t_A and t_B required by each robot to do a movement step. Now we can analyze the total time required as a function of t_A and t_B as follows:

- Case 1 - Without considering the time required to do a task: The total time needed to explore the whole area is the time required by the slowest robot:

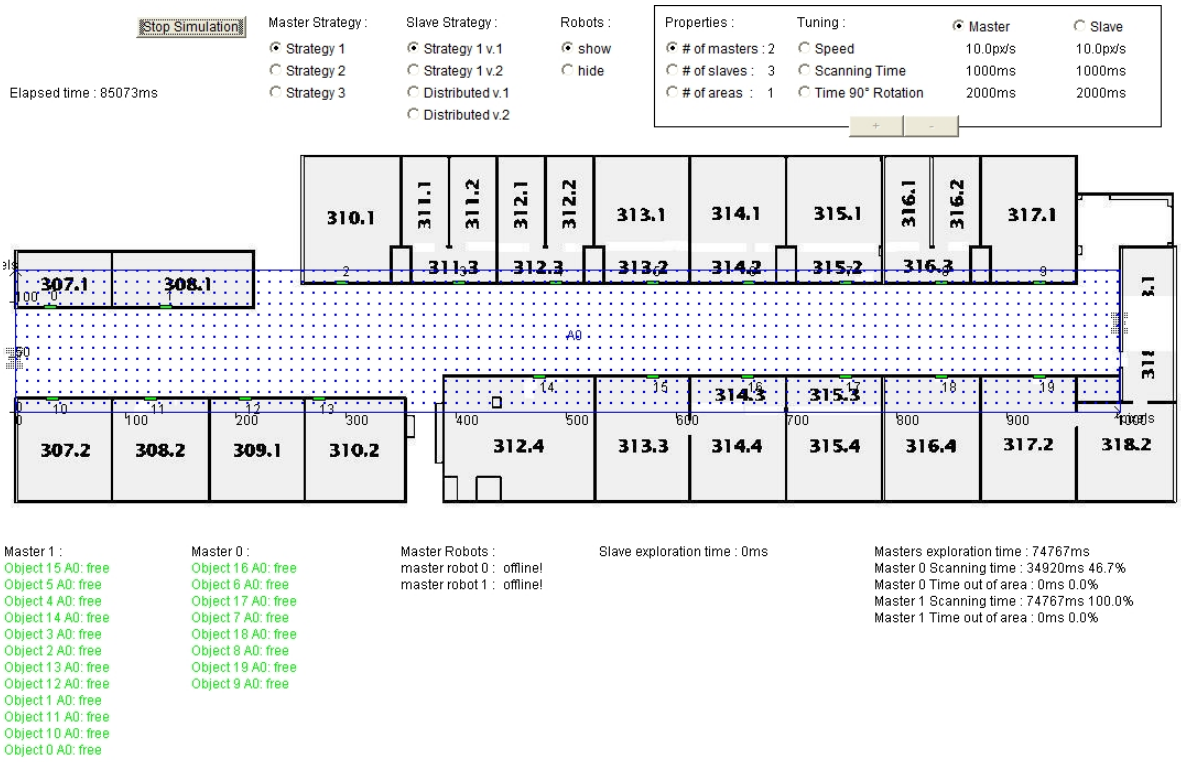
$$time = max\left\{\left(t_A * N_A\right), \left(t_B * N_B\right)\right\}$$

  where N_A and N_B are the number of movement steps necessary to explore the sub-area of the first and second master respectively.

- Case 2 - Taking into consideration the time of completing a task: Naturally the time required to complete a task depends on the task itself and the sensor used to do this task. Within our work, the master performs a landmark extraction at each movement step in order to be used in the localization process. The processing time of such a task should be taken into account, because the time required to extract natural landmarks



(a) (b) (c) (d) (e)



(f)

Figure 5.10: Steps of the frontier strategy: (a) an environment model to be explored, showing the start position of the robots. (b), (c), (d) and (e) the robots start navigating through their sub-areas in different direction to explore the environment. In (d) and (e) the first master has finished exploring its sub-area and locating the features and could not help the second robot since there is no cooperation at this stage. (f) both robots have finished their task showing the information that results from this exploration strategy.

is considerably high and affects the total time of exploration. Thus, the total time needed to explore the whole area depends on two factors: the speed of a robot, and the time needed to extract a feature. Hence, the total time needed to explore the whole area is as follows:

$$time = max\left\{(t_A * N_A + t_{task} * (N_A + 1)), (t_B * N_B + t_{task} * (N_B + 1))\right\},$$

where t_{task} is the time required to do a feature extraction at each movement step.

5.4.2 Sub-areas Strategy

The limitation of the first strategy is that it applies the prospect function only in the directions of the explorer, which sometimes causes one robot to finish the exploration in its direction quickly, but it cannot go to help the other robot to finish the exploration. From this limitation the idea of the second strategy is driven. It is called sub-areas strategy.

Within this strategy, only a limited area is to be selected by any robots. When it finishes the current area, the robot selects a new area as a second move. Each time, it applies the prospect function on the available areas. If the robot finishes its region in its direction, it can help the other in order to finish the exploration quickly. So, the idea of this method is to split the area into n sub-areas (instead of only two), and both robots have to cooperate in order to explore these sub-areas efficiently.

If there is no cooperation at all, then each robot will explore all the sub-areas because they can not find out which areas have already been explored by the other robot. Thus, the time of the non-cooperative exploration can be calculated as follows:

$$time\,(noncooperative) = max\left\{t_A, t_B\right\} * N_{total}$$

where t_A and t_B are the time required by the first and second master to do a movement step respectively, and N_{total} is the total number of movement steps necessary to explore the whole area.

However, we would like to have an optimal use of multiple robots, thus a cooperation must be performed between the robots in order to get an efficient time of exploration. This is exactly the idea behind the sub-areas strategy: When a robot selects a sub-area, it sends a message to the other. Using these messages, it is possible to know which sub-areas have already been explored and which areas are still unexplored. So, the robot moves to a new region, which has not been explored by both robots yet. During the exploration, the robot selects a new sub-area iteratively each time that it finishes its own one, until the whole area is explored.

In the contrary of the first strategy in which the robots cooperate only at the beginning deciding the direction of the exploration process, the sub-area strategy illustrates an implicit cooperation aspect. The cooperation consists in sharing the information about the already explored sub-areas and in dynamically allocating more sub-areas to the quickest robot rather than the slowest one, depending on their real required time.

Figure 5.11 shows a possibility of applying the sub-areas strategy in the general case of splitting the area in n sub-areas. These areas are to be known at the exploration time. The strategy takes into account the navigation cost from one sub-area to another.

Within this strategy, it is not guaranteed that both robots finish at the same time. In general this is not the case, the quickest robot receives in most cases more sub-areas than the slowest one. The task is finished when the last part of the whole area is explored. Three experiments were conducted in which the robots were set up with different speeds. In the first experiment, both robots had an equal speed, and in the second experiment the first master had a higher speed than the second one, whereas in the third experiment the first master had twice the speed of the second one. Figure 5.12 presents the final result of the exploration at each situation. As a result, both of them explore approximately the same number of sub-areas in the case of equal speed. However, in the third experiment the first master explores more sub-areas than the second one. It can be noticed from the figure 5.12(c) that the first master explores 5 sub-areas marked with a blue points, whereas the second master explores only 2 sub-areas. Furthermore, the first master has changed its direction and comes from the back side of the corridor to the front side of it to help the second one.

In order to expect the total time of the exploration within this strategy, we suppose that the area is divided into n equally complex sub-areas based on the assumption that each robot takes only a limited area in each movement step. So, each sub-area is completely explored in $\frac{N_{total}}{n}$ steps. It means that the whole area can be explored within:

$$time = max \left\{ \begin{array}{c} \left(t_A * \left(p * \frac{N_{total}}{n}\right) + t_{task} * \left(p * \frac{N_{total}}{n}\right) + 1\right), \\ \left(t_B * \left((n-p) * \frac{N_{total}}{n}\right) + t_{task} * \left((n-p) * \frac{N_{total}}{n}\right) + 1\right) \end{array} \right\},$$

where p is the total number of sub-areas explored by the quickest robot, and t_{task} is the time needed to extract a feature.

5.4.3 Dynamics Strategy

In the two strategies explained before, the cooperation occurs either at the beginning of the exploration process as in frontier strategy, or if a robot starts a new sub-area as in the sub-areas strategy. During the exploring of any sub-area, there is no cooperation. Another possibility would be to make the robots cooperate at each time a change in the status of the task or event happens. The main idea of the third strategy consists in a dynamical update of the size of the sub-area assigned to each robot, where the name "dynamic strategy" origins in this process.

In this strategy, there is an update each time a new event occurs. Possible events in our task are either finishing to explore a sub-area or extracting a natural feature. Each time that such an event occurs, the robot in question sends a message to inform about that event and the size of the sub-area reserved to each robot is updated depending on the content of the message.

This method tries to find out an approximation of the optimal partitioning of the area so that both robots are finished at the same time. If an update does not deliver the optimal

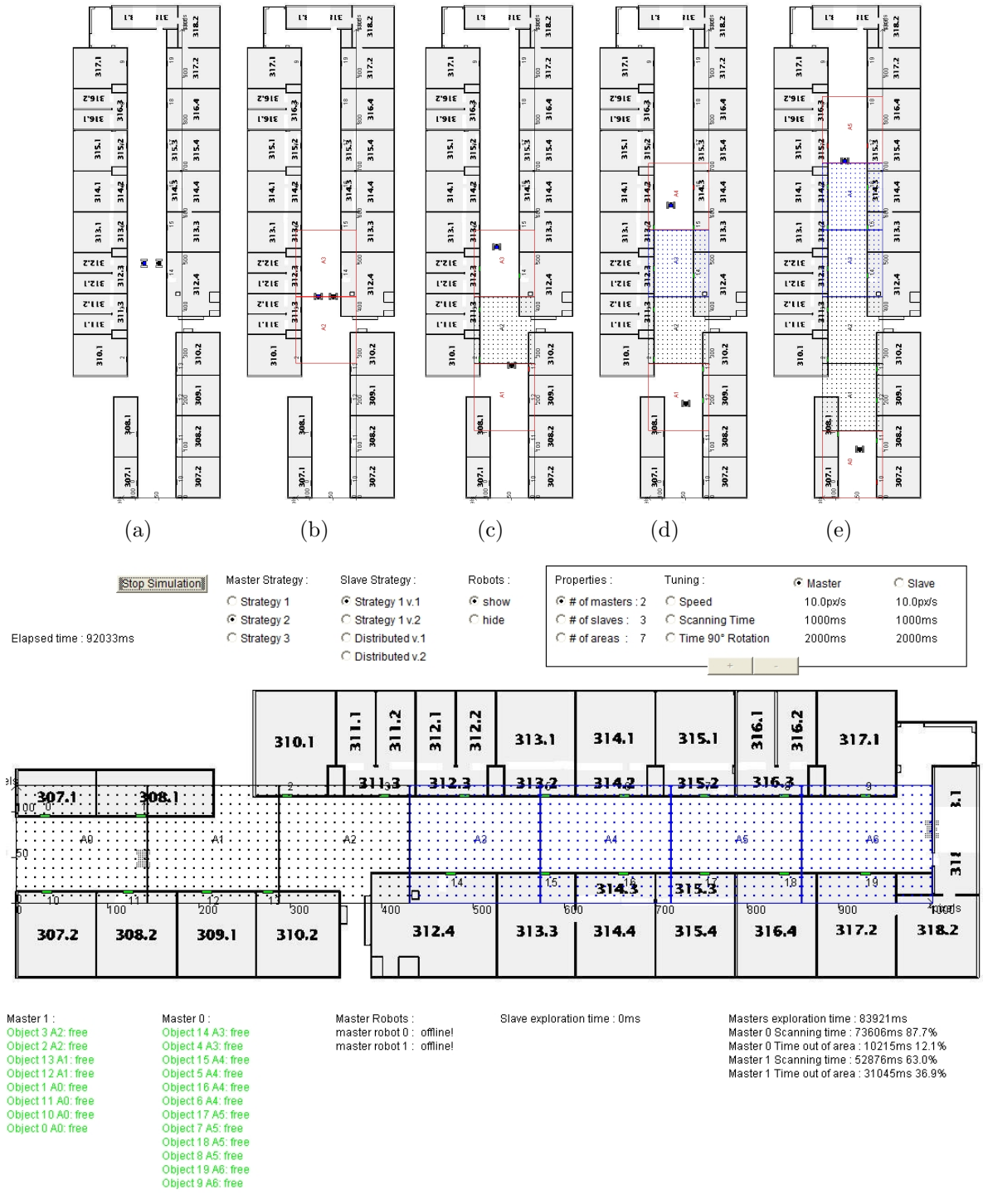
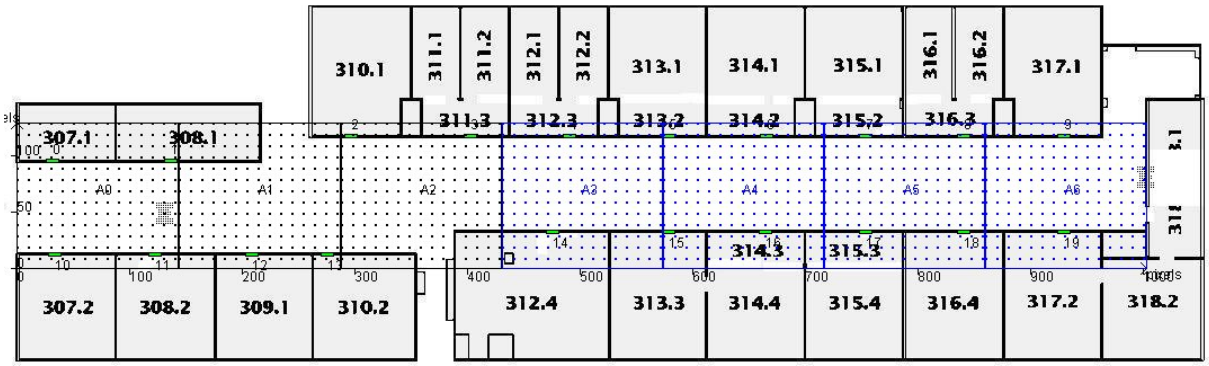
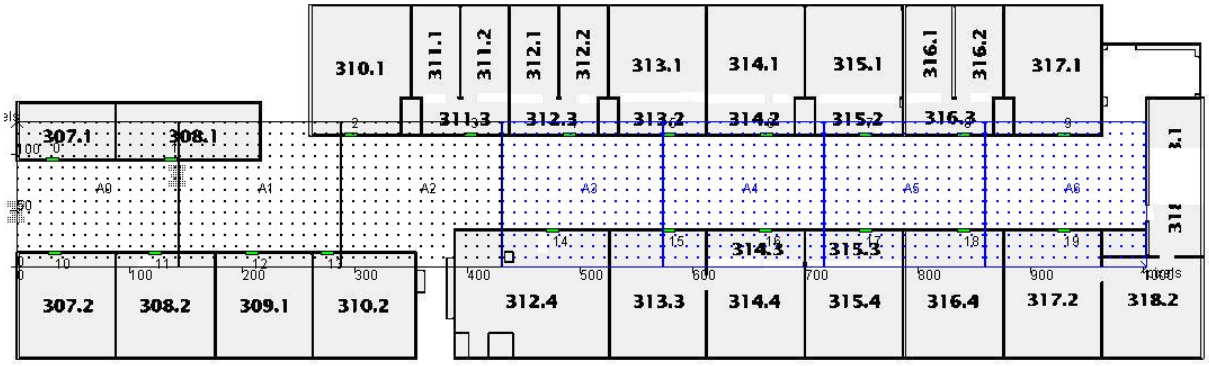


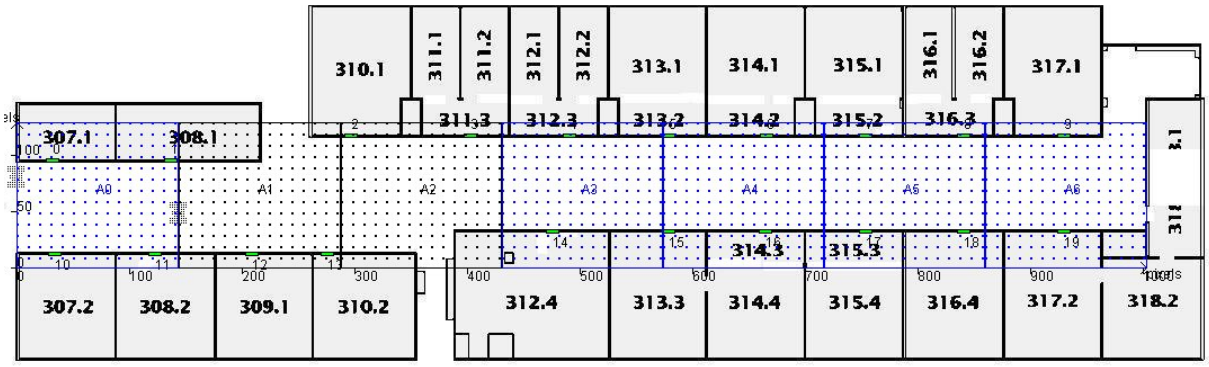
Figure 5.11: Steps of the sub-areas strategy: (a) an environment model to be explored, showing the start position of the robots. (b) Each robot starts, reserving in a cooperative manner a limited sub-area for itself, and (c) the second master has finished its sub-area and reserves a new sub-area to be explored by it. In (b), (c), (d) and (e) the robots navigate through their sub-areas to locate any features. (f) Both robots have finished their task. The figure also shows the final partition of the sub-areas completed by each robot, the first master has finished exploring 4 sub-areas, whereas the second master has explored only 3 sub-areas.



(a)



(b)



(c)

Figure 5.12: Analysis of the movement steps of the sub-areas strategy using different speed: (a) both robot have equal speed. (b) The first master has a higher speed than the second one, within this experiment the first master has finished exploring its 4^{th} sub-area and navigates to other side of the environment to help the slowest one, but the slowest robot has reserved the last unexplored area before the first master gets there. (c) The first master has twice the speed of the second one. The first master has finished exploring 5 sub-areas, whereas the second one has only finished 2 sub-areas.

partitioning at any stage, the next update will correct it, and so on until the whole area is explored. Within this strategy, it is guaranteed that the use of the robots is as optimal as possible, the navigation time without performing the exploration task is minimal, and the time in which the robots are idle is minimal as well.

Figure 5.13 illustrates the case of splitting the area in two sub-areas. The update of the size of the sub-area took place when an event stated before occurred to either of the two robots. As shown in figure 5.13, the robots start from opposite corners of the area in order to dynamically update the size of the sub-area. The benefit from this strategy is only achievable when the robots start from the opposite corner of the explored sub-area.

Thus, this strategy could be used in an application where part of the environment is known, having two opposite entrances, and the required task is to find some targets of interest or to extract some natural features of this environment.

In order to analyse the time required to complete an exploration with this strategy, we have to know how this strategy is working. The goal of this strategy is to let both robots finish at the same time, reducing the time of navigation as much as possible, and thus resulting in a minimum idle time for both robots.

Within this strategy, when a robot stops to extract a feature, it sends a message to inform about it. Automatically, the size of its area is updated so that a part of its area is assigned to the second robot. Furthermore, when a robot finishes its own sub-area, it also sends a message, but this time a part of the area of the other robot will be assigned to it. Thus, the size of the new sub-area allocated to the less busy robot depends on the following criteria:

1. Position of the robot in its area.

2. Position of the last feature extracted by the other robot. This information is taken from the last message received from the other.

Theoretically, the analysis of the time required to achieve this strategy cannot be estimated a priori, since the properties of the area and the locations and number of features to be extracted are unknown at the beginning. So, we estimated only the time based on the last update.

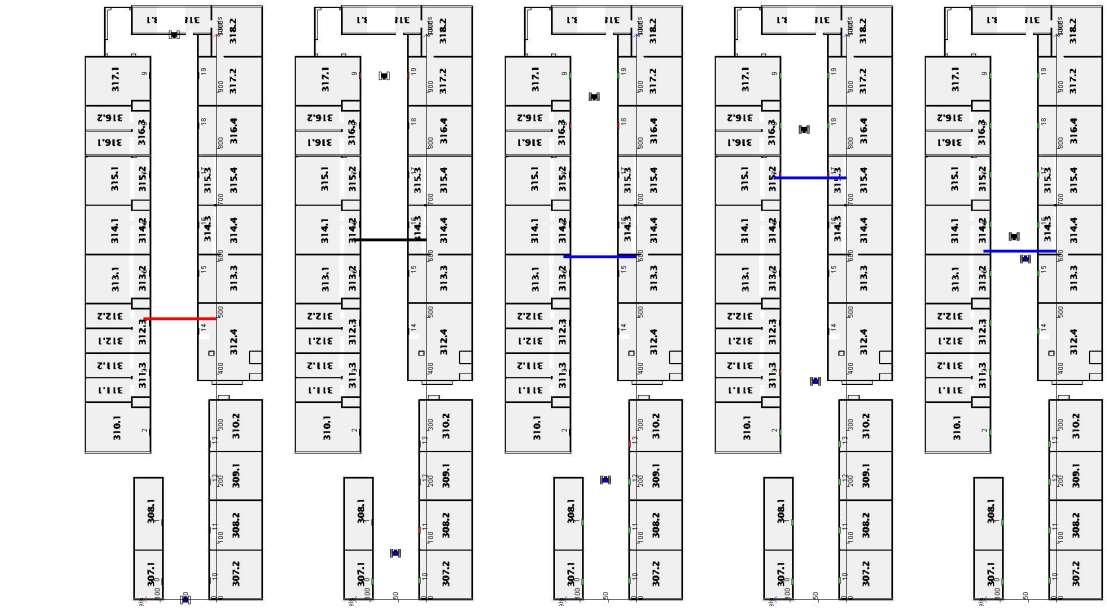
Suppose at the last update that the first master has already performed N_A movement steps to explore its sub-area, then the second master is performed $(N_{total} - N_A)$ movement steps to explore its sub-area, where N_{total} is the total number of movement steps necessary to explore the whole area. So, we have after the last update that:

$$t_A * N_A + t_{task} * (N_A + 1) = t_B * (N_{total} - N_A) + t_{task} * ((N_{total} - N_A) + 1)$$

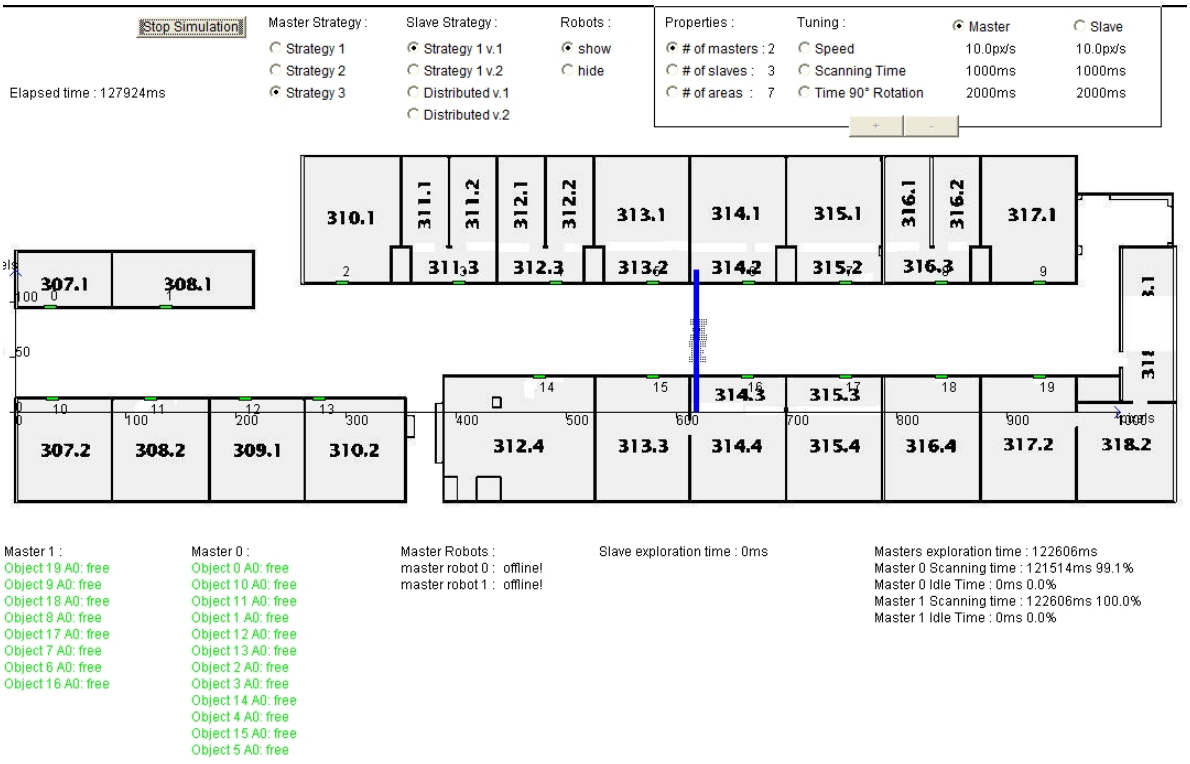
where t_A and t_B are the time required by the first and second master to do a movement step respectively, and t_{task} is the time needed to extract a feature.

This equality is given because the robots are supposed to finish at the same time. Thus, the time required to achieve this strategy is as follows:

$$time = t_A * N_A + t_{task} * (N_A + 1)$$



(a) (b) (c) (d) (e)



(f)

Figure 5.13: Steps of the dynamic strategy: (a) an environment model to be explored, showing the start position of the robots and the logical partitioning of the area by a red line. (b) The second master has found an object and sends a message to inform about it, an update of the size of the areas takes place. (c), (d) and (e) the update process is done by the first master. (e) Both robots at the step before the last step, and they do approximately finish at the same time. (f) The final result of the strategy, the blue vertical line represents the final partitioning of the area after the last update. It is completely different from the initial one and is an approximation of the optimal partitioning of the area for both robots.

5.4.4 Discussion

According to the classification mentioned in section 5.1, our masters worked cooperatively and thus are classified as strongly coordinated distributed multiple robots. This is because each one of them explored a different area at the same time autonomously, applying some prospect values in order to coordinate themselves more efficiently.

Three factors are used in order to compare the three strategies mentioned before, namely cooperation degree, navigation time, and the exploration time. The factor cooperation degree describes how much both robots are cooperative in order to achieve the task efficiently. Within the frontier strategy, the robots are cooperative only at the beginning in order to choose different directions of exploration. From the coordination point of view, this is an optimal solution. However, we noticed from figure 5.14 that the first master has finished exploring early and it can not try to help the other in order to finish the exploration quickly. This limitation is improved within the sub-areas strategy, in which the robots are cooperating each time when the exploration of a sub-area is finished, such that the degree of cooperation is improved. Thus, we classify this strategy to be of a medium degree of cooperation. The idea of the dynamic strategy is to let the robots cooperate any time an event occurs such as the detection of a feature or the completion of a sub-area. Therefore, the robots strongly cooperate during the exploration process. Thus, we classify this strategy to be a strategy with a high degree of cooperation.

We now consider the second factor, namely the navigation time. This is the time required by a robot to move from one place to another without performing any exploration task. From figure 5.14, it can be noticed that the second master has a high navigation time in the case of the sub-areas strategy. This is due to the fact that the second master was moved from the back to the front side of the environment in order to help the other master. This limitation cannot be avoided in an unknown environment.

The third factor concerns the efficiency of each strategy. We have found that the dynamic strategy has a better exploration time, the best cooperation degree, and the best partitioning of the whole area, because the robots are cooperating more often than in the other strategies. However, it can be applied only to certain applications with a partially known environment. On the other hand, the sub-areas strategy is more suitable in the general case, where the partitioning of the whole area into sub-areas improves the computational and storage requirement of the exploration process.

5.5 Cooperative Strategies – Slave Case

In section 5.4, we presented cooperative strategies using two masters. The coordination process between them is easily achieved by the fact that each master explores a different region simultaneously. On the other hand, each master has a team of n slaves, such that the coordination process among them will be more complex, and more attention is necessary to achieve an efficient coordination.

Two strategies are used to evaluate the system performance by using n slaves. Within the first strategy, each master sends its team to certain places in the environment in order

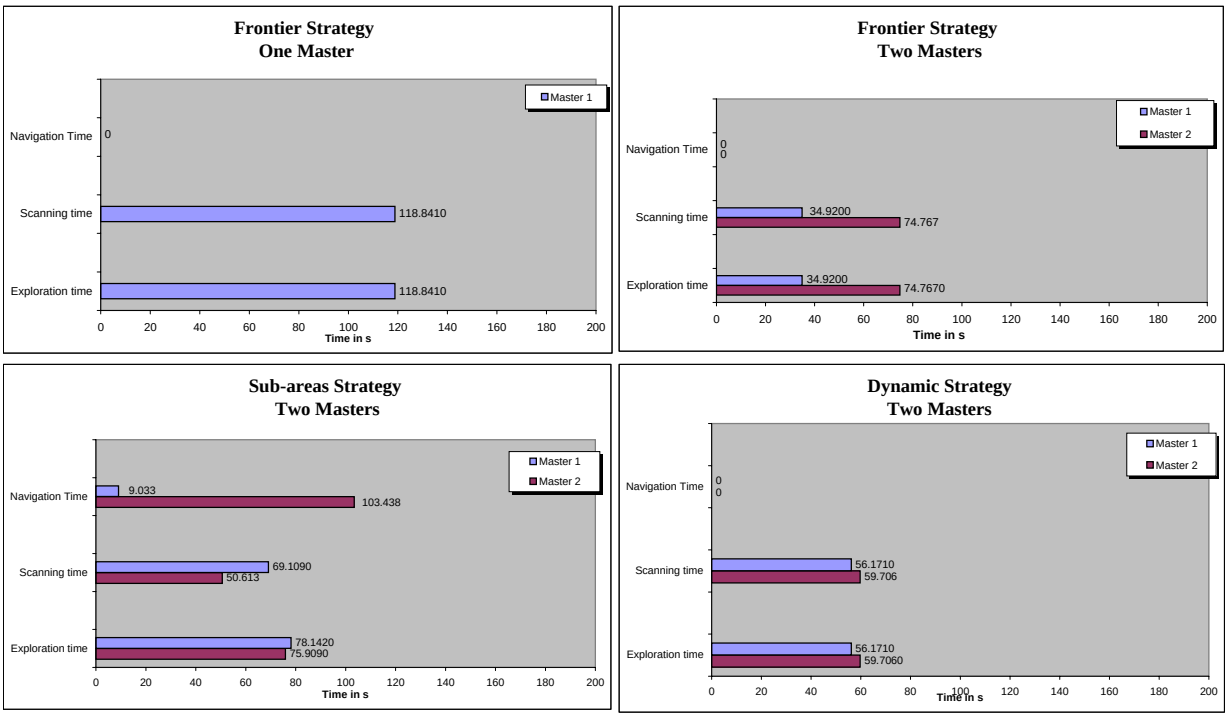


Figure 5.14: The performance of master cooperative strategies: (a) and (b) frontier strategy, (c) sub-areas strategy, and (d) dynamic strategy.

to get a 3D scanning of this place. Each master prepares a list of tasks $\{TaskList_{Slave_1}, TaskList_{Slave_2}, \cdots, TaskList_{Slave_n}\}$ to each slave of its team $\{Slave_1, Slave_2, \cdots, Slave_n\}$ taking into account the coordination mechanism and position of each slave of its team. This is called *centralized strategy*. In this strategy, the slaves only perform the planning, navigation and scanning process to achieve these tasks autonomously.

On the other hand, the second strategy gives a higher degree of autonomy to the slaves. In this strategy, the slaves receive a list of tasks, and they should choose by themselves which slave carries out which task, taking into consideration some coordination mechanism and the position of the other slaves. Besides the planning, navigation and scanning process, in this strategy the slaves also perform the coordination ability in order to achieve the tasks sent by their master. This strategy is called *distributed strategy*. To study the performance of these strategies, we have used the environment model presented in section 5.4.

5.5.1 Centralized Strategy

The benefit from a master-slave architecture is that the slaves are coordinated by a master in an efficient way, knowing the position of its team. In the contrary to master strategies in which only two masters are cooperating to perform an exploration task, the centralized strategy uses n slaves to achieve the 3D scanning of some places defined by their master in a centralized form. That means, a master is fully responsible for the coordination of its team.

Thus, the master has to perform some processes knowing in advance its slaves position and the places of the tasks to be carried out by the slaves in order to coordinate its team

efficiently. The term efficiency here means that the slaves should be distributed over the environment in a way to avoid task conflicts and sensor interferences.

To achieve these requirements, the master performs the coordination among its slaves as follows:

1. Order the tasks into a list according to the time of its task detector, in which the task was explored first is being the first member of this list.
2. Order the slaves into a list based on their position with respect to the starting point of the exploration done by the master, in which the nearest slave to this point is being the first member of the list.
3. Define how many tasks should be assigned to each slave.
4. To assign a task to any slave, the master should in this stage check the sensor range of this slave in order to avoid the sensor interference.

Algorithm 5.2 illustrates the use of these steps more clearly. The result of this algorithm is a list of tasks $\{L_{Slave_1}, L_{Slave_2}, \cdots, L_{Slave_n}\}$ to be carried out by each slave. This algorithm was tested using in simulation up to 20 slaves and it could be proved that it works efficiently.

Figure 5.15 illustrates the centralized strategy in case of splitting the tasks among three slaves. The slave have to plan and then navigate through the corridor in order to achieve the tasks demanded by their master. The slaves perform the tasks in an order taking into the account the navigation cost to each target region, which gives a good performance when achieving these tasks. It can be noticed from figure 5.15(c), 5.15(d) and 5.15(e) that the slaves are distributed over the environment efficiently and perform the scanning of different places simultaneously.

Within this strategy, the communication takes places only between the master and its team. There are no communication protocols exchanged among the slaves themselves, which reduces the degree of cooperation. However, the cooperation is performed as a whole team with respect to their master. It is can be noticed from figure 5.15(f) that the yellow slave scanned 6 places, the blue one scanned 7 places and the black one also scanned 7 places. In the process of communication between the master and its team, the information is exchanged at the beginning, at which time a list of tasks to be done by the slave is sent to each one. The slave starts achieving these tasks, and sends the result to its master as soon as it finishes a task, from which the master can get the acknowledgement that the slaves have already carried out another of their task.

5.5.2 Distributed Strategy

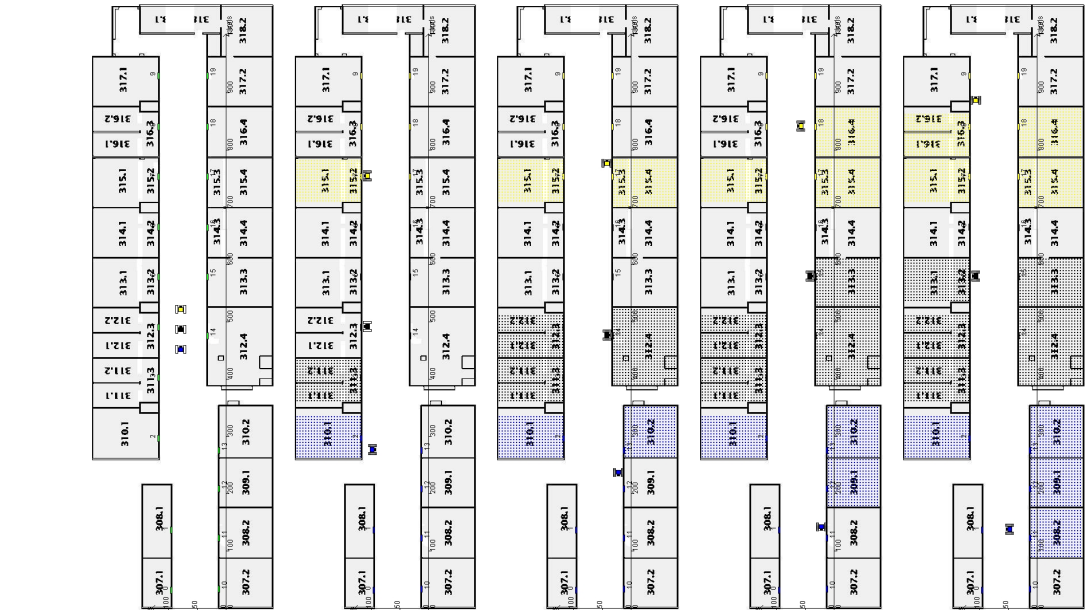
The primary objective of this strategy is to develop the fundamental capabilities that enable n slaves to coordinate themselves over the environment and to distribute tasks among them in an efficient way in order to improve the system performance. The basic concept is to enable individual slaves to act independently, while taking into consideration a more tight and precise coordination as the coordination which was achieved by the centralized strategy.

Input: $TaskList, SlaveList$ /*List of all tasks to be done by the slaves*/
/*The slaves list*/

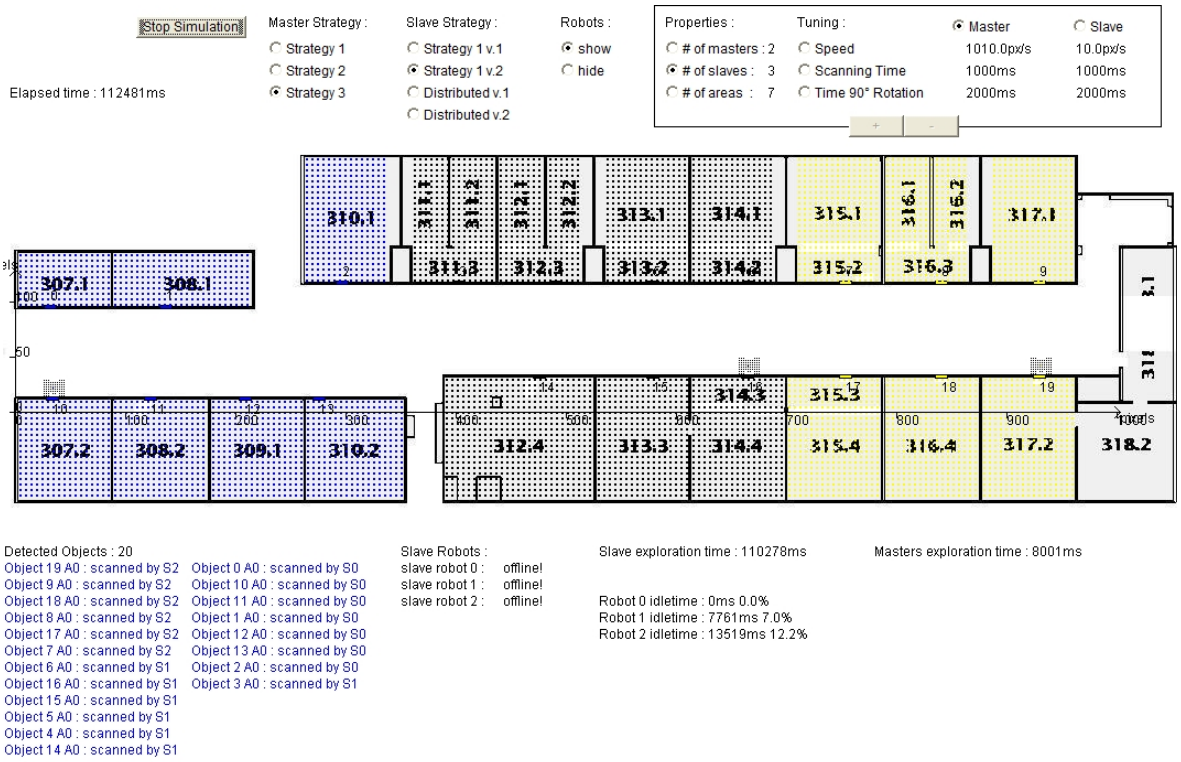
Output: $L_{Slave_1}, L_{Slave_2}, \cdots, L_{Slave_n}$
/*List of the tasks to be done by each slave*/

1: $ThresholdTask = \frac{Numberoftasksinthelist}{Numberofslavesinthelist}$
/*Calculate the actual number of tasks to be done by each slave*/
2: $RestTask = module\left(\frac{Numberoftasksinthelist}{Numberofslavesinthelist}\right)$ /*If number of slaves is an odd number*/
3: **while** *the slave list is not empty && the task list is not empty* **do**
4: **for** *all tasks in the list* **do**
5: Get Left and Right Tasks; /*A coordination tool*/
6: **end for**
7: **for** *all slaves in the list* **do**
8: Get Left and Right Slaves; /*A coordination tool*/
9: **end for**
10: **for** *Both Slaves* **do**
11: $TaskNo. = ThresholdTask$
12: **if** $No.ofslavesinthelist \leq RestTask$ **then**
13: $TaskNo. = ThresholdTask + 1$
14: **end if**
15: $L_{Slave} = Task$ /*Add this task to the list of the slavetasklist*/
16: Delete this task from the task list
17: $TaskNo. = TaskNo. - 1$
18: CheckCoordination()
/*Find other tasks within sensor range of the slave in the task list w.r.t this task*/
19: **if** *found* **then**
20: $mTask = Numberoftasksfoundintheneighbour$
21: **else**
22: $mTask = 0$
23: **end if**
24: **while** $mTask \neq 0$ **do**
25: $L_{Slave} = Task$
26: Delete this task from the task list
27: $TaskNo. = TaskNo. - 1$
28: **end while**
29: **while** $TaskNo. \neq 0$ **do**
30: $L_{Slave} = Task$ /*Assign a nearest task to the slave task list*/
31: Delete this task from the task list
32: $TaskNo. = TaskNo. - 1$
33: **end while**
34: delete slave /*Remove this slave from the list*/
35: **end for**
36: **end while**
37: **Return** $L_{Slave_1}, L_{Slave_2}, \cdots, L_{Slave_n}$

Algorithm 5.2: The outlet of the coordination algorithm done by the master in order to perform the centralized strategy among n slaves.



(a) (b) (c) (d) (e)



(f)

Figure 5.15: Steps of the centralized strategy with 3 slaves: (a) an environment model to be explored, showing the start position of 3 slaves. In (b), (c), (d) and (e) the slaves start navigating through the environment in different directions to perform the scanning tasks, taking into consideration the navigation cost. In (c), (d) and (e) it can be noticed that the three slaves are distributed very well over the environment. (f) All slaves have finished their tasks showing all the information related to this strategy.

Individual slaves will be more autonomous than the slaves in the centralized strategy, yet will be able to negotiate with one another to perform tasks in efficient and quick ways.

As opposed to the centralized strategy in which the master is fully responsible for the coordination of its team, the distributed strategy gives a higher degree of autonomy to the slave team.

However, the master-slave structure is still necessary, as it gives the system some vital information to perform the distribution strategy. For example, a list of tasks and the position of each slave of the team need to be known.

In the centralized strategy, no information is exchanged among the slave team in order to achieve the tasks. The prospect values presented in chapter 4 are not used by the slaves as well, since only the master uses them to distribute the tasks among its team.

On the other hand, the distributed strategy allows the slaves in one team to exchange some information in order to achieve a certain degree of coordination. This is achieved by applying the prospect values by each slave in order to accomplish its tasks more efficiently.

To achieve these requirements, the master should perform the following preliminary steps:

1. Form a list of groups, in which the tasks detected in the same step of exploration are inserted in one group.

2. Sort these groups into a list according to the time stamp of its task detector, in which the group which was explored first is the first member of this list.

3. Sort the slaves into a list based on their position with respect to the starting point of first group to explore, in which the slave nearest to this point is the first member of this list.

4. Send this information to all slaves in its team.

At this stage, each slave then performs the coordination process in order to coordinate itself with the other slaves based on the information received from their master. Thus they achieve the coordination taking into consideration the following steps:

1. Find how many tasks should be achieved by itself.

2. To reserve a task, the slave should at this stage check its sensor ranges in order to avoid any sensor interference.

3. All the slaves should follow a similar way to work on the list of task groups to predict exactly which task is taken over by which slave.

4. As soon as a task is reserved by any slave, it should send an acknowledgement to the other slaves to be taken into account when they choose their tasks.

Algorithm 5.3 illustrates of these steps more clearly. The result of this algorithm is a list of tasks to be carried by this slave. This algorithm was tested in simulation using up to 20 slaves and it could be proved that it works more efficiently than the centralized strategy.

Figure 5.16 illustrates the distributed strategy in the case of splitting the tasks among three slaves. In the contrary to the centralized strategy, the slaves in the distributed strategy have to coordinate themselves first, to assign some tasks, to plan and then navigate through the corridor in order to achieve the tasks sent by their master. The slaves perform their tasks taking into account the navigation cost to achieve each task. It can be noticed from figures 5.16(c), 5.16(d) and 5.16(e) that the slaves are distributing themselves over the environment efficiently and perform the scanning of different places simultaneously. They are also using the sensor range more efficiently than in the first strategy, this can be noticed clearly from figure 5.16, where this fact is represented by a circle defining the sensor range of each slave.

5.5.3 Discussion

According to the classification presented by Farinelli [Farinelli 04], within the distributed strategy the slaves themselves are working cooperatively and thus are classified as strongly coordinated distributed multiple robots. On the other hand, the master with its slaves in the centralized strategy are classified as strongly coordinated weakly centralized multiple robots.

Three factors are used in order to compare both strategies, namely the degree of autonomy, the time of exploration and the number of slaves necessary to get an optimal performance. Within the centralized strategy, each slave receives a list of tasks to be carried out by it and the slave has to perform by itself the planning, navigation and scanning process. In some applications it can be necessary to have a slave which can perform its tasks individually. We have implemented the distributed strategy which takes this fact into account.

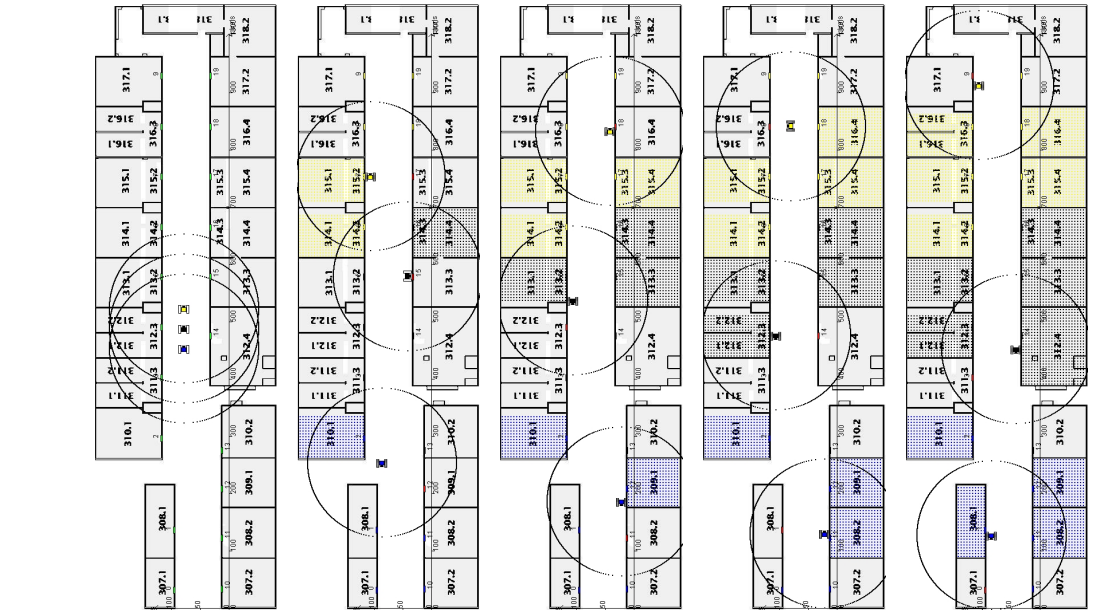
The distributed strategy reaches a very good system performance compared with the centralized one. However, the time of performing the scanning process in the distributed strategy is a little bit higher due to the fact that each slave should, besides the planning, navigation and scanning process, perform the coordination process in order to coordinate itself among several slaves.

One more factor which has an effect on the system performance is how many slaves should be used to achieve the task more efficiently. Different experiments were performed in simulation using up to 20 slaves in two different areas. It could be proved that a higher number of slaves results in a better system performance. However, this also depends strongly on the environment in question. For example, in the first building, more than 9 slaves are useless because the system performance will not be improved any more, whereas we found in the second building that more than 6 slaves are useless.

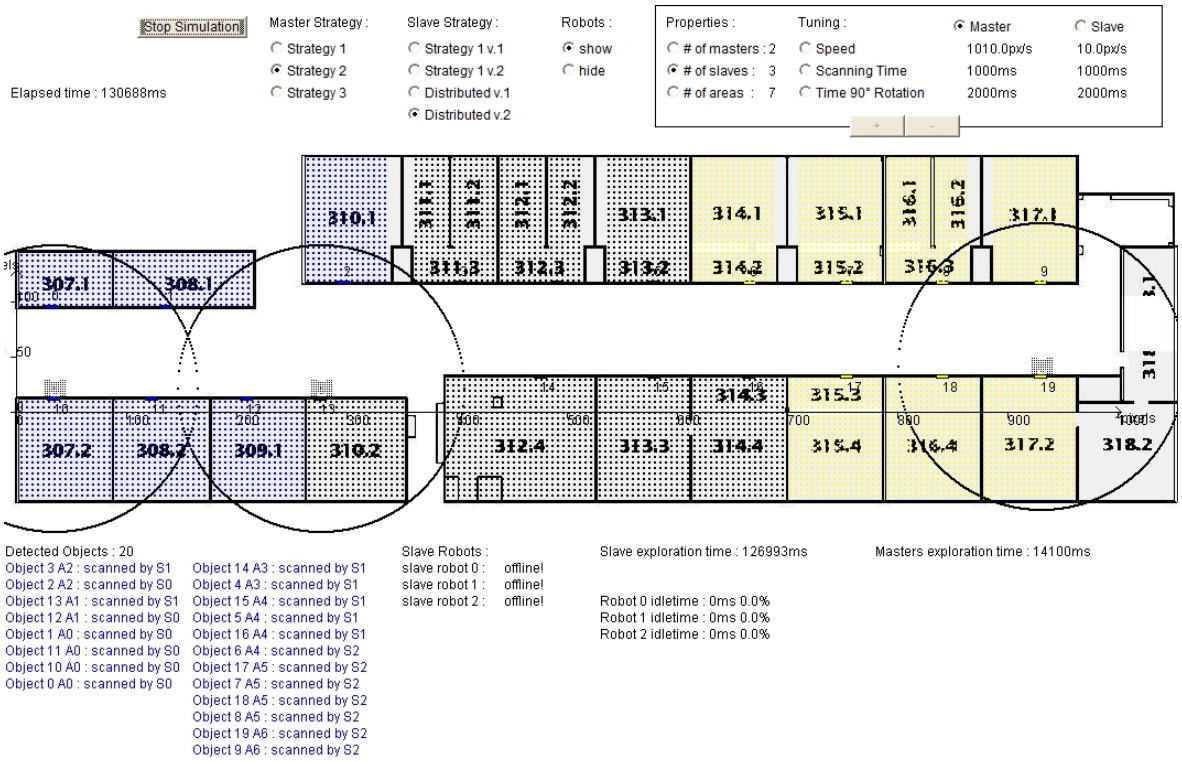
Input: $TaskgList = \{group_1, group_2, \cdots, group_m\}, SlaveList$
/*Tasks and slaves are ordered from left to right according to their position*/
Output: L_{Slave} /*The task list to be done by this slave*/

```
1: Calculate ThresholdTask and RestTask
2: n_Slave = No.of slaves in the list          /*To be used in the rest task calculation*/
3: Slaveworkingqueue ← SlaveList               /*Form a working queue for tasks and slaves*/
4: Taskworkingqueue ← TaskgList
5: while Slaveworkingqueue is not empty && Taskworkingqueue is not empty do
6:   Get Left and Right Group Tasks and Corresponding Slaves; /*A coordination tool*/
7:   for Both Slaves do
8:     TaskNo. = ThresholdTask
9:     if n_Slave ≤ RestTask then
10:      TaskNo. = ThresholdTask + 1
11:    end if
12:    repeat
13:      if No.of the task in this group ≤ TaskNo. then
14:        groupTask ← No.of the tasks in this group
15:      else
16:        groupTask ← TaskNo.
17:      end if
18:      while groupTask ≠ 0 do
19:        if Slave(Loop)==ThisSlave then
20:          L_Slave = task
21:        end if
22:        delete task;                        /* remove this task from this group*/
23:        TaskNo. = TaskNo. − 1
24:      end while
25:      if there is still some tasks in this group then
26:        pushback task;   /*Push back this task as a new group into the task group*/
27:      end if
28:      if TaskNo. ≠ 0 then
29:        mTask = TaskNo.
30:        GetNextGroupTask                    /* pop up a next group task*/
31:      else
32:        mTask = 0
33:      end if
34:    until mTask = 0
35:    if Slave(Loop)==ThisSlave then
36:      Return L_Slave                        /*The slave has already detected all its task*/
37:    end if
38:    n_Slave = n_Slave − 1
39:  end for
40: end while
```

Algorithm 5.3: The outlet of the coordination algorithm done by each slave in order to perform the distributed strategy.



(a) (b) (c) (d) (e)



(f)

Figure 5.16: Steps of the distributed strategy executed by 3 slaves: (a) an environment model to be explored, showing the start positions of 3 slaves. (b), (c), (d) and (e) The slaves start navigating through the environment in different directions to perform the scanning tasks, taking into consideration the coordination factor and the navigation cost. In (c), (d) and (e) it can be noticed that the three slaves are distributing themselves very well over the environment. (f) All robots have finished their tasks, showing all the information related to this strategy.

5.6 Conclusion

This chapter is concerned with the cooperative strategies in the master-slave architecture in order to get a better system performance. It started with an overview about cooperative robotics and communication, and with presenting a classification of multiple robot systems focused on coordination. According to this classification, our master-slave architecture is classified as cooperative, strongly coordinated and weakly centralized multiple robot system. The type of communication among the robots was presented as well, namely active and passive communication, which are used to achieve the cooperation task.

The second section introduced the idea of active communication, in particular concerning the data exchange among the robots. A description of how data is exchanged among the robots was presented focused only on the cooperation scheme among them. Two different types of data are exchanged, namely the position of the robots and the maps of certain places. An identifier was used to identify the type of the position or the maps to be sent by the robots. For example, edge, occupancy grid, or feature-based maps are exchanged among the robots.

The third section gave a close lock of how passive communication was performed. Two different approaches were discussed to present the idea of the passive communication, namely visual and laser detector. With both approaches, the robot tries to estimate its position based on the position of a stationary robot and some color object or artificial beacons.

In fourth section, three cooperative strategies were performed using two masters, the frontier strategy, the sub-areas strategy and the dynamic strategy. It could be shown that the dynamic strategy has a better system performance. However, it can be applied only with a certain application with partially known area. On the other hand, the sub-areas strategy is more suitable to our work. In this strategy, the partitioning of the whole area into sub-areas improves the computational and storage requirements.

Within the fifth section, two cooperative strategies were performed using n slaves, namely the centralized strategy and the distributed strategy. The ideas and results of both strategies were presented, and the performance of each strategy was evaluated. It was also illustrated in detail which algorithms are performed by the master and which algorithms by the slaves. It was found that the distributed strategy gives a good system performance compared with the centralized strategy.

Chapter 6

Feature Extraction Techniques

6.1 Introduction

Navigation is one of the most fundamental competences of any mobile agents. It can generally be defined as controlling motion to arrive at a known location. In the case of mobile robots, navigation can be decomposed into the sub-problems of position estimation, path planning and local motion control. Hypotheses of the vehicle position and orientation is knowing in the recently literatures as SLAM problem. The hypotheses are commonly derived from an interpretation of sensor signals provided by range sensors, beacon detectors or computer vision. It is common to maintain an estimate of the position and orientation using odometry. Because of incorrigible drift errors, odometry-based navigation such as path integration can only be used over very short distances, and robot navigation systems that operate over realistic distances of several hundred meters route length have to use other methods for localization and path planning. The most common approach on such a navigation scale is to use perceptual landmarks. As a result, the core of the landmark-based SLAM is the landmark extraction task.

In order to let the mobile robot be able to sense its location, navigate its way toward its destination, and avoid obstacles it encounters using a landmark-based representation, these landmarks have to be extracted as reliably and precisely as possible. Such a process remains difficult to implement because of measurement noise, scene clutter and dynamic objects.

In mobile robotics, one way to acquire position information is by computer vision. Vision-based systems are attractive because they are self-contained, in the scene that they require no external infrastructure such as beacons or radio station. Vision-based systems in principle can operate indoors and outdoors, virtually everywhere as long as there are rich visual features for place recognition. An example are stereo vision sensors, which have become the most popular sensors for navigation and mapping as they directly provide texture information about the environment by using some image processing techniques. In 3D space, these textures need to be identified and classified in order to extract a specific feature to be used in the navigation process. The indoor space scene analysis allows to detect natural structures relevant for navigation like doors in stereo images of the environment.

Another way to acquire position information is by range finder sensors, such as the RoSi laser scanner. They directly provide distance measurements at high data density in heading

direction and at lower data density at the boundaries. In 3D space, dense 3D point clouds can be being complex. The triangulation of these 3D points as surface models provides a means of reducing this complexity while at the same time keeping the information content. The texture of a scene can be extracted by using a digital camera mounted above the RoSi sensor. These textures can be further processed to find some useful and reliable features to be used in the area of autonomous navigation.

Segmentation, the partitioning of sensor data into meaningful sections or regions with respect to a particular application, is an important first step in the feature extraction analysis. With data available as image vision data or range image data, the region shape can be analyzed as a key for appropriate labelling of the region in the data analysis step. In data compression, the regions form a basis for a compact representation of the image data. The quality of the prerequisite image segmentation is a key factor in determining the level of performance of most of these image analysis and data compression approaches.

The segmentation is based on measurements taken from the image and might be based on grey level, color, texture, depth or motion. In particular, color image segmentation has been the subject of different research activities in the robotic field. For example, Rano and his group [Rano 01] presented an attempt to use a colored pixel segmentation for mobile robot navigation and localization. Wasik and Saffiotti [Wasik 02] used a robust color segment which is applied in the RoboCup robotic application. Rous et al. [Rous 05] used two cameras to identify doors based on Hough transform techniques and the homogenous color of a feature.

The color image segmentation is a process of extracting one or more connected regions from the image data satisfying a homogeneity criterion which is based on features derived from spectral components. These components are defined in a chosen color space model. A connected region in the image can be detected and interpreted using some classification technique to identify the region as a known object. Most research on vision-based feature identification systems has concentrated on identifying specific objects in a scene (e.g. [Lee 04] and [Rous 05]). In their work, the segmentation process has been augmented by some additional knowledge about the objects in the scene such as geometric and color properties.

6.1.1 Color Spaces

A color model is a 3D unique representation of a color. There are different color models and the choice for one of them is purely problem-oriented. For instance, the color model RGB (Red-Green-Blue) is used in hardware applications like PC monitors, digital cameras, or scanners, whereas the color model CMY (Cyan, Magenta, Yellow) is used in color printers. In color image segmentation, the two models widely used are Hue-Saturation-Value (HSV) and $L^*u^*v^*$ color models.

In the scene analysis, the appearance of real objects is determined by some properties like the texture, the lighting conditions, and the shade effects. The pixel colors affects strongly as well. Therefore, the choice of a suitable color model is crucial in the image processing or in scene analysis.

RGB Color Space

A digital camera generally delivers color images via an RGB color model; thus a particular pixel is associated with a three-dimensional vector (r, g, b) which provides the respective color intensities. When all three of these colors are at their maximum, the pixel appears white. When all three are at the lowest, the pixel appears black. Various combinations of the three vector elements result in a large number of distinct colors and shades. For example, equal portions of red and green without any blue result in a shade of yellow. Figure 6.1 represents the RGB model as a cube. Each point in the cube represents a specific color.

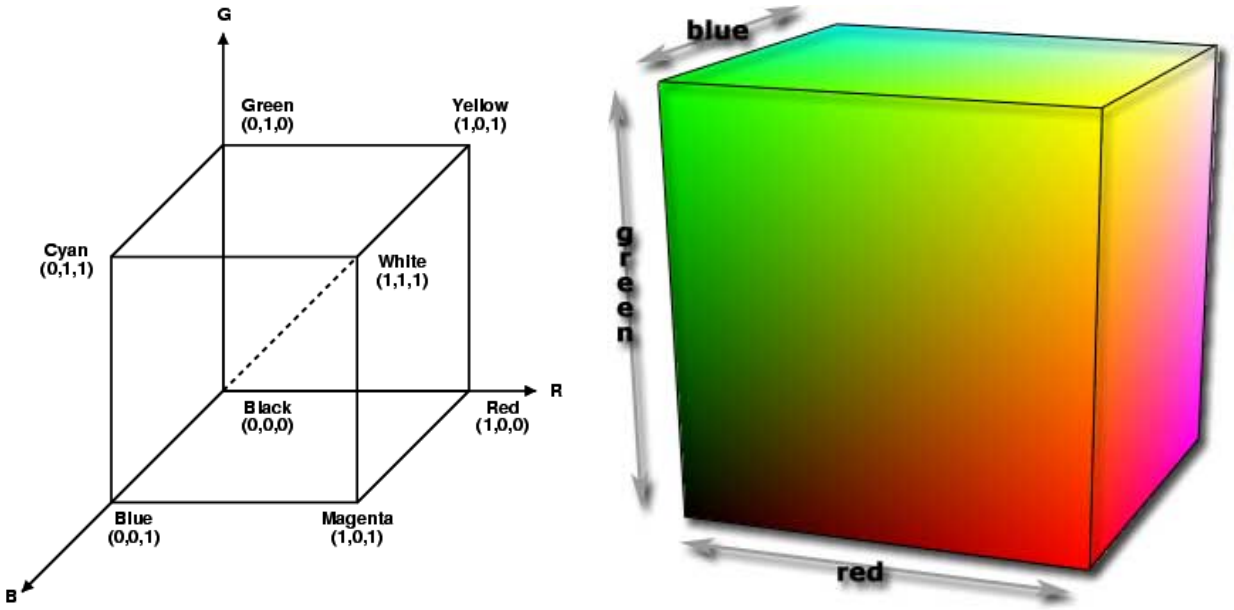


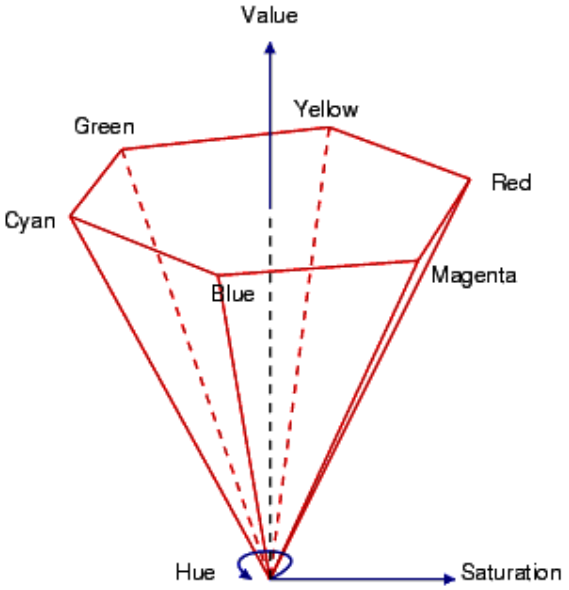
Figure 6.1: (a)The schema of the RGB color cube with black at 0,0,0, white at 1,1,1, and primary and complimentary colors on the vertices. The dotted line represents the shades of grey, and (b) the RGB color cube itself

HSV Color Space

The RGB model is a familiar color space often used in image processing, but it suffers from an important drawback for many robotic vision applications. In such applications, the classification of colored features in the RGB model is not robust due to the variations in the brightness of illumination. That is because of conical volume relation of the RGB model, which can not be represented with simple threshold. In contrast, the HSV model have the advantage that it separates out the intensity (luminance) from the color information (chromaticity), since chrominance is generally less sensitive than luminance to illumination changes [Feyrer 99]. Thus a particular color can be described as a column spanning all intensities. This type of color space is therefore more useful than RGB for robotic applications.

The HSV color space has a hexcone shape as illustrated in figure 6.2, which is based on polar coordinates. The hue is defined as an angle in the range $[0, 2\pi]$. The saturation represents the purity of the color and is measured as a radial distance from the central axis with values between 0 at the centre to 1 at the outer surface. The central vertical axis represents the

intensity [Shapiro 94]. For example, for a given intensity and hue, if the saturation is changed from 0 to 1, the perceived color changes from a shade of grey to the most pure form of the color represented by its hue.



H	S	V	Color
0	1.0	1.0	Red
120	1.0	1.0	Green
240	1.0	1.0	Blue
*	0.0	1.0	White
*	0.0	0.5	Gray
*	*	0.0	Black
60	1.0	1.0	
270	0.5	1.0	
270	0.0	0.7	

Figure 6.2: HSV Color model: (a) the HSV hexcone, and (b) representation of HSV values

$L^*u^*v^*$ Color Space

In image processing, it is of particular interest to have a perceptually uniform color space where a small difference in a component value is approximately equally perceptible across the range of that value. The color model such as RGB is far from uniform. Thus, the $L^*u^*v^*$ model was introduced, where L is a lightness scale from 0 (black) to 100 (refenece white), u is a red-green scale and v is a yellow-blue scale. As a result, $L^*u^*v^*$ is a perceptually uniform color space in which calculated distances between colors in the color space approximately reflect visually perceived color differences. Figure 6.3 shows a typical example of such a representation. The color image in figure 6.3 left is mapped onto the three-dimensional $L^*u^*v^*$ color space.

6.1.2 Segmentation Methods

Most image segmentation approaches can be divided into three groups [Sonka 03] which are based on: edge detection [Chapron 92], region growing [Liu 94], or histogram analysis [Comaniciu 97].

Histogramming techniques are sometimes referred to as clustering techniques. They have been applied to the segmentation of colour images, as they identify homogenous clusters of points in the feature space (such as the RGB colour space, or the HSV colour space, etc.) and then label each cluster as a different region. The homogeneity criterion is usually that of color similarity, i.e. the distance from a cluster to another cluster in the color feature

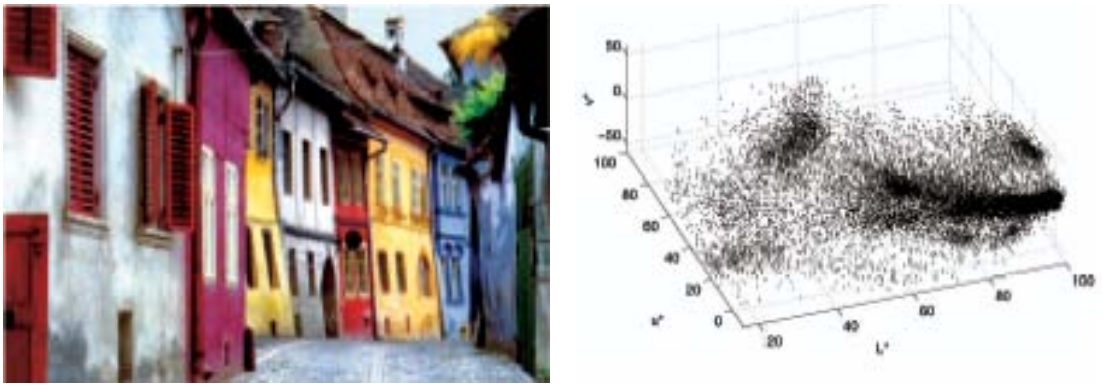


Figure 6.3: Example of representing $L^*u^*v^*$ color model: (a) a 400x276 color image, and (b) corresponding $L^*u^*v^*$ color space[Comaniciu 97]

space should be smaller than a threshold. The disadvantage of this method is that it does not exploit spatial information, and thus ignores information that could be used to enhance the segmentation results.

Boundary or edge detection methods are generally based on a local analysis of the image signal variation in order to extract the lines which represent the boundaries of regions, they do not necessarily produce closed, connected region boundaries. The major inconvenience of these methods is that they require a preliminary model of the world to allow the grouping of the extracted edges into coherent objects. Furthermore, edge detection on noisy, complex image data often produces missing edges and extra edges that cause the detected boundaries to form a bad representation of the connected regions.

Region growing approaches to segmentation use a global methodology which consists in selecting an image pixel around which the corresponding region will expand. The pixel choice must be made according to an optimum criterion, and sometimes this choice may require a preliminary edge extraction. This problem of the pixel selection is avoided with split and merge algorithms. Within these algorithms, the original image is divided into basic regions. Adjacent patches are merged into one region if they satisfy a similarity criterion. Results are strongly dependent on the initial splitting and on the merging criterion. As a result, the region growing approaches exploit spatial information and guarantee the formation of closed connected regions. They will therefore be employed in the course of our work. Additionally, the mean shift algorithm as a form of clustering technique will be in the evaluation.

6.2 Region Growing Segmentation

6.2.1 Definition

As its name implies, region growing is a procedure that merges pixels or sub-regions into larger regions based on predefined criteria, which satisfies the objective of the segmentation. The following formal definition is taken from [Sonka 03]:

Let R represent the entire image region composed of a finite set of regions $R_1, R_2, \ldots, R_N$ as a result of the segmentation process that partitions R into N sub-regions, which satisfy

the following conditions, where N is the total number of segmented regions in an image, and $H(R_i)$ is a binary homogeneity evaluation function of the region R_i:

1. $R = \bigcup_{i=1}^{N} R_i$
   This condition indicates that the segmentation must be complete; which means that every pixel must be assigned to a region.

2. $R_i \cap R_j = \phi$ for all i and j, $i \neq j$
   The second condition indicates that the regions must be disjoint.

3. $H(R_i) = True$ for $i = 1, 2, \ldots, N$
   The third condition deals with the prior knowledge that must be satisfied by the pixel in a segmented region; that is, for example $H(R_i) = true$ if all pixel in R_i have same color level.

4. $H(R_i \cup R_j) = False$ for $i \neq j$ R_i adjacent to R_j
   Finally, this condition indicates that regions R_i and R_j are two different regions in the scene.

6.2.2 Region Growing Process

The region growing algorithm consists of 3 steps:

1. During the pre-processing, the initial seed points are determined.

2. The seeds are selected and regions grow around.

3. Insignificant regions are rejected or merged.

The first task in the pre-processing step is to determine the initial seed points. When there is no prior knowledge about the scene, these points are randomly selected. In our exemplary environments, pixels of the relevant features tend to have a predefined hue value. Based on this information, we select all pixels satisfying this requirement as starting points.

The second step is to choose criteria for region growing. In our work, we choose three criteria for a pixel to be allocated to a region:

1. The color blobs between any pixel and the seed have to be matched, satisfying that the absolute difference between them must be less than a defined threshold.

2. The pixel has to be connected to at least one pixel in any region, in order to allocate it to that region, which sometimes yields a merging process if a pixel is connected to more than one region.

3. The growing process should take into account the region size and the ratio between the height and width of the bounding region. This assumption is based on a prior knowledge of the dimension of the features we search for, and it is used in the third step.

Finally in the third step, once the region growing of the second step is finished, an over segmentation process is used to improve the segmentation result. Due to some noise in the original image or a wrong choice of the seeds small regions appear. The post-processing task either merges these regions into a bigger one that satisfies the connectivity constraints or deletes them from the regions list if the growing conditions are not satisfied.

The process of these steps is summarized in the pseudo code of algorithm 6.1, as it forms the basis of the feature extraction algorithm described in section 6.4. Figure 6.4 shows a simple example of growing a region by following the steps mentioned in the algorithm 6.1. At the pixel point $p_0\,(u_0, v_0)$, the process begins the growth in the raw image date, starting with the assumption that this pixel forms a region. This region almost certainly do not satisfy the assumption (4) of the formal definition in subsection 6.2.1, and so the region will be merged as long as assumption (3) of the in formal definition remains satisfied. The process stops if no more regions can be merged maintaining the aforementioned assumption.

```
Input: Img                                                    /*Color Image of the scene*/
Output: R                                                     /*Region list*/
 1: while unclassified pixel exists do
 2:    Img(s) ← seed point
       /*select pixels having a predefined value based on a prior knowledge as a seed point*/
 3:    R ← 0 + Img(s)                                         /*initiate the region model*/
 4:    ϵ ← Threshold                                          /*select a threshold value*/
 5:    Img(c) ← Img(s)                                        /*set the current pixel to a pixel s*/
 6:    RecursiveGrowing (Img, Img(c), R)
 7: end while
 8: Return R
    /*************************************************************************/
 1: Function RecursiveGrowing(Image img, Pixel img (c), Region R)
 2: for each unclassified pixel img (q) in the neighbourhood of img (c) do
 3:    if (‖c − q‖ ≤ 1) && (|img (q) − img (c)| < ϵ) then
 4:       R ← R + img(q)                                      /*add a pixel to the existing region*/
 5:       img(c) ← img(q)                                     /*adjust the current pixel*/
 6:       RecursiveGrowing (img, img(c), R)
 7:    end if
 8: end for
 9: Return
```

Algorithm 6.1: The region growing algorithm

As a general conclusion, the region growing procedure uses the HSV color model to identify prior knowledge about the feature to be extracted. During the segmentation process, the conditions stated in subsection 6.2.1 should be satisfied in order to get a robust segmentation result. The three steps of the algorithm were explained, starting with determination of the initial seed regions depending on a prior knowledge of the feature color. The regions then grow around these initial seeds. Finally, a post-processing is required in order to reject or merge insignificant regions depending on a prior knowledge about the dimensions of the feature to be found.

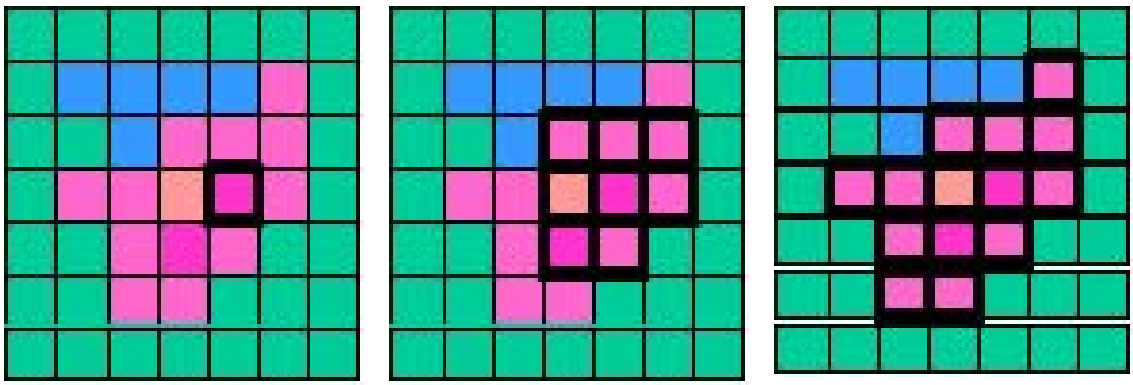


Figure 6.4: Steps of region growing segmentation

This method guarantees a continuous segmentation in closed, connected regions, because it considers both the color information and spatial details. Furthermore, the processing time is considerable good, the algorithm can thus be applied online in the field of robotics. The algorithm implementation does not have high memory requirements and is easy to implement under the MCA environment framework. However, the region growing approaches suffer from their inherent dependence on the selection of a seed region, which sometimes lead to wrong segmentation results. One more disadvantage of this method is that it needs some prior knowledge to improve the segmentation process which implies that the segmentation process is not completely autonomous.

6.3 Mean Shift Segmentation

As we saw from the previous method of segmentation, a prior knowledge about the feature in question is required in order to get a robust feature extraction by region growing. As a result of looking for a segmentation method that does not require any prior knowledge, we used the mean shift segmentation method.

The mean shift segmentation algorithm belongs to the family of segmentation algorithms based on clustering techniques as explained in subsection 6.1.2. Mean shift clustering is a *non-parametric* density estimation technique based on kernel density estimation, which does not require prior knowledge of the number of the clusters, and does not constrain the shape of clusters. In the next subsection, the mean shift procedure is introduced. We then present in subsection 6.3.2 how the mean shift procedure is applied in order to segment a colour image.

6.3.1 Mean Shift Procedure

As Comaniciu and Meer proposed [Comaniciu 02] and [Comaniciu 97], mean shift segmentation is a feature space analysis concept, with the aim to estimate cluster centres in a specific feature space. The mean shift segmentation algorithm is based on a statistical tool called mean shift, which assumes that the feature space can be regarded as an empirical probability density function (p.d.f) of the represented parameter. Dense regions in the feature space

correspond to local maxima of the p.d.f, that is, the modes or peaks of the unknown density. Figure 6.5 presents this idea.

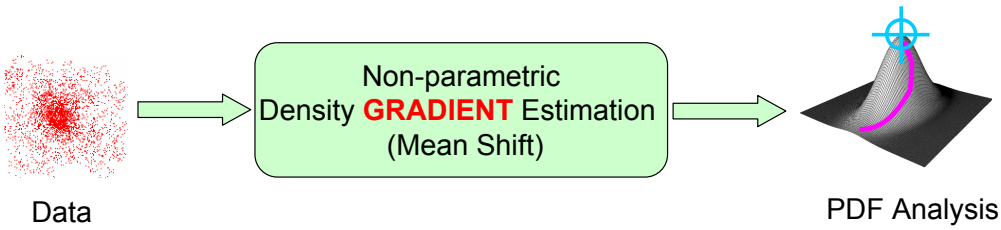


Figure 6.5: Statistic definition of the mean shift

Once the location of a mode or a peak is determined, the cluster associated with it can be delineated based on the local structure of the feature space. Thus, the mode detection is an important part of the feature space analysis. In the mean shift segmentation algorithm, such a mode detection process is based on the mean shift procedure.

Based on the work of Comaniciu and Meer [Comaniciu 02], the pseudo code of the mean shift procedure is illustrated in algorithm 6.2. The procedure is a successive iteration procedure, which iterates two steps until the convergence criterion is satisfied. Firstly, mean shift computes its associated peak by defining a spherical window at the data point of radius r and computing the mean of the points that lie within the window. Secondly, the procedure then shifts the window to the mean. The mean shift procedure is guaranteed to converge at a nearby point where the density estimator has zero gradient [Comaniciu 97], that is a mode. This is shown graphically in figure 6.6 where the initial position of the kernel window is chosen at a random location far from the mode of the data and the window is moved by a magnitude equal to the mean shift vector at the current window location in each step. It can been seen that the mean shift vectors gradually converge as the window moves near the maximum density region.

Input: $P_i := \{(x_i, y_i), (L_i, u_i, v_i)\}$
/*Feature space (spatial + colour)*/
Output: mode c_i
/*The mean location in the search window*/
1: $W_{size} \leftarrow$ Window Size /*Determine the window size (usually small)*/
2: **for** *each point* P **do**
3: $w_i \leftarrow d(P, P_i)$ /*Compute the Euclidean distance between them*/
4: $\vec{m} = \sum_{i \in window} w_i(\vec{P_i} - \vec{P})$ /*Compute a weighted mean of the shift in the window*/
5: $c_i \leftarrow P + m$
6: **if** $(|c_i - P| \leq Threshold)$ **then**
7: **Return** c_i /*If converged then return the centroid of the data*/
8: **end if**
9: $P \leftarrow c_i$ /*Centre the search window at the mean location computed at step 5*/
10: **end for**
11: **Return** c_i

Algorithm 6.2: Mean shift procedure

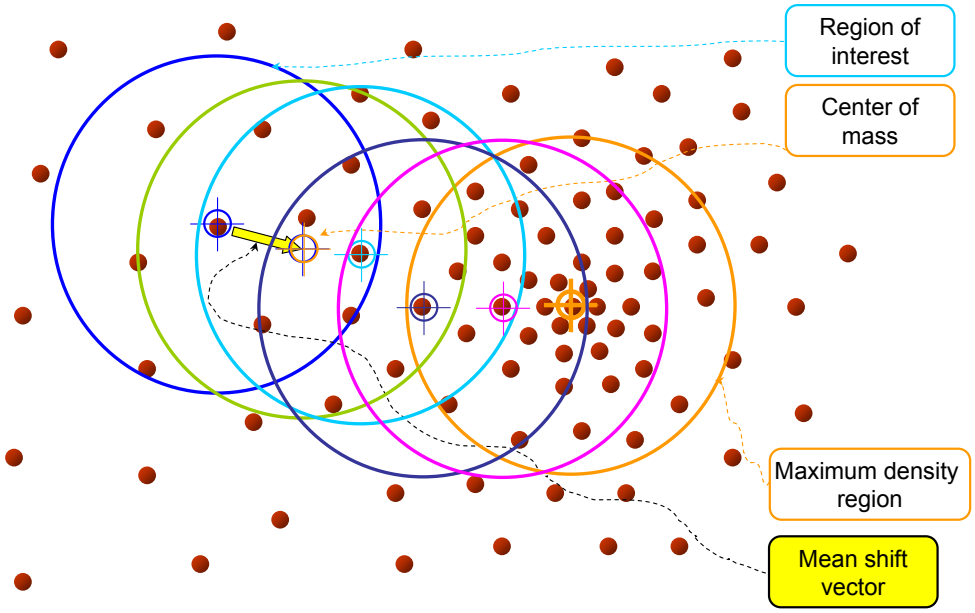


Figure 6.6: Mode detection using mean shift procedure

The set of all locations that converge to the same mode defines the basin of attraction of that mode. Thus, the delineation of the clusters is a natural outcome of the above mode detection process. After convergence, the data points visited by all the mean shift procedure converging to that mode, automatically delineates a cluster of arbitrary shape.

6.3.2 Feature Space Analysis

The algorithm provided by Comaniciu and Meer robustly determines the cluster means based on a feature space analysis and relies on the mean shift procedure. A feature space is a space of feature vectors. As shown in figure 6.7 , the input to the algorithm is spatial information of pixel in the image and the colour space $L^*u^*v^*$ of this image for each pixel, which are represented as a point in the feature space. Within the algorithm, a Euclidean distance metric is used. Unfortunately the Euclidean distance in RGB colour space does not correlate well with the perceived difference in colour by human. For this reason, the nonlinear $L^*u^*v^*$ colour space is used. In this space, the Euclidean distance models the perceived difference more efficiently.

In their research work, Comaniciu and Meer state a simple, adaptive steepest ascent mode seeking algorithm. The idea of this algorithm is summarized in the flowchart in figure 6.7. The algorithm starts by computing the mean shift window location for each initial position. Then, it merges windows that end up on the same peak or mode. Finally, the data which these merged windows traversed is clustered.

The algorithm is implemented under our MCA framework environment. The results of this algorithm at two different positions in the indoor environment are shown in figures 6.8 and 6.9. The resolution of the colour images is 640×480. As a result, the method performs the segmentation process autonomously without any prior knowledge of the feature. It

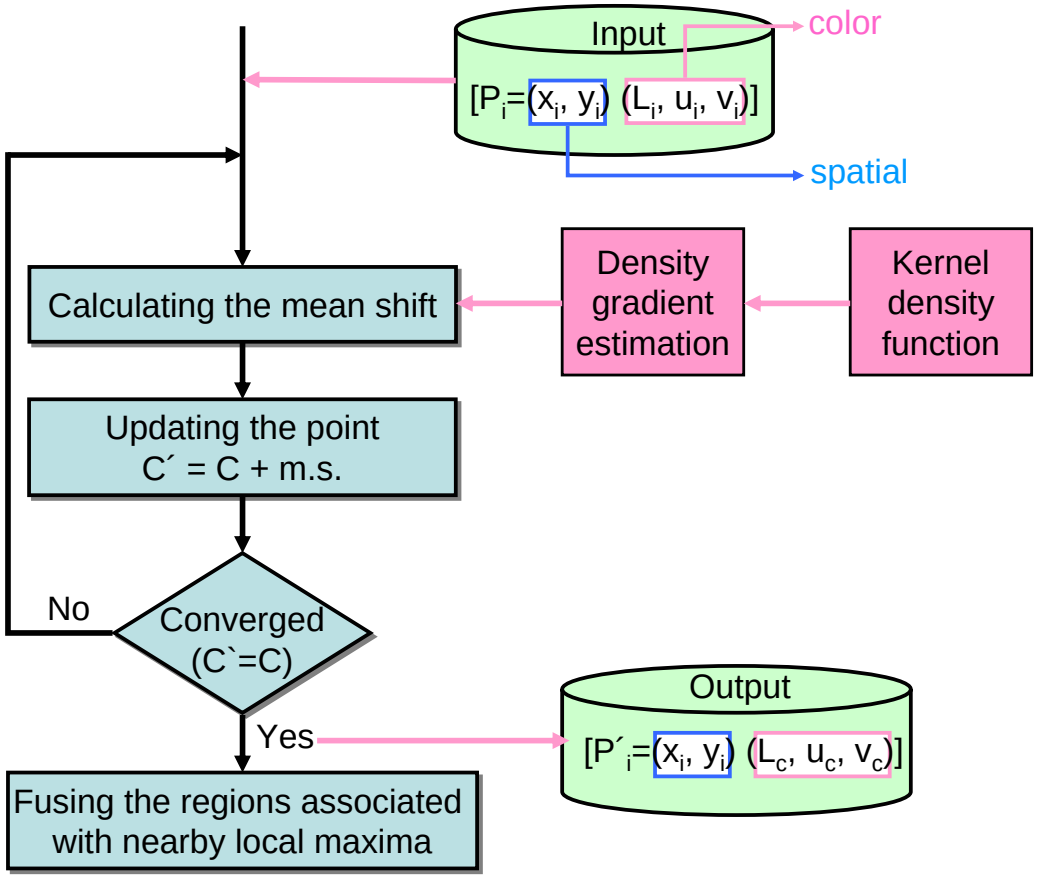


Figure 6.7: Flowchart of the mean shift segmentation method

achieves robust results. However, it has two disadvantages. Firstly, the computing time is too high, approximately four times more than the time needed by the region growing method. Secondly, the memory requirement is too high as well, which sometimes leads to an unstable execution of the algorithm. For these reasons, we did not select this method as a core of the feature extraction algorithm, as is not executable online. That means, in its current form it cannot be applied in the field of robotic exploration.

6.4 Concept of the Feature Extraction Algorithm

The extraction of natural landmarks from indoor environments is an extremely well studied topic in mobile robotics. Davison in [Davison 02] detected a relative location of point features using a stereo camera. He used the operator of Shi and Tomasi to select regions of high interest and to represent the features as 15×15 pixels patches which are used in the correspondence process. In his work, there is no feature validation algorithm applied at the initialization step.

Other researchers like [Lee 04] and [Rous 05] detected specific landmarks (e.g. doors), which are topological structures of indoor environments. For example, Rous et al. [Rous 05] used two cameras to identify the doors, starting their algorithm with a Hough transform generating convex polygons. Polygons with similar homogenous colors are segmented and

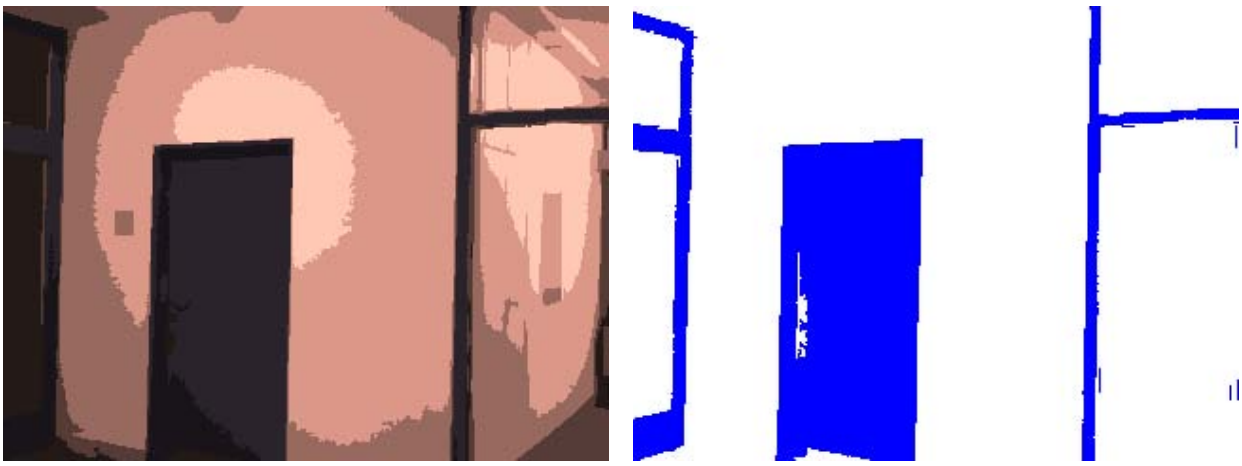


Figure 6.8: Result of the mean shift segmentation method: (a) Image of a door, and (b) the desired result of the scene analysis should be a separation of merged regions like doors

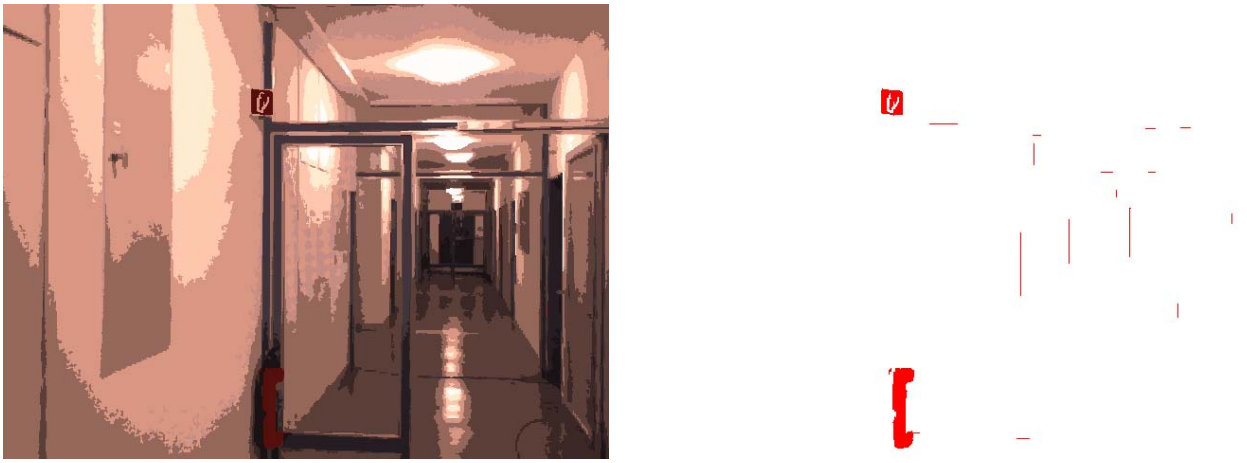


Figure 6.9: Result of the mean shift segmentation method: (a) Image of a hallway, and (b) the desired result of the scene analysis should be a separation of merged regions like a fire extinguisher

merged by a region growing process and then assigned to known objects. Within this work, a good execution time of the algorithm is obtained, and it is therefore applicable to mobile robot navigation. But it is necessary that the whole image of the door is acquired.

Lee et al. [Lee 04] implemented a door detection algorithm using a simple PC camera. They tried to overcome the fact that the image does not cover the complete door by detecting some other features of the door itself. However they could not show the robustness of their algorithm.

Palmer [Palmer 99] has introduced a theoretical approach to vision by decomposing human visual perception at the algorithmic level into a sequence of four basic stages:

1. Extracting the image structure.
2. Recovering the surface in depth.
3. Describing 3D objects.

4. Identifying objects in terms of known categories.

The definition of each stage is based on a different kind of output representation and on the processes that are required to compute it from the input representation. Figure 6.10 illustrates these four stages which Palmer calls *image-based*, *surface-based*, *object-based* and c*ategory-based* perception stages. These four stages provide a fairly general and robust framework for understanding vision as a computational process.

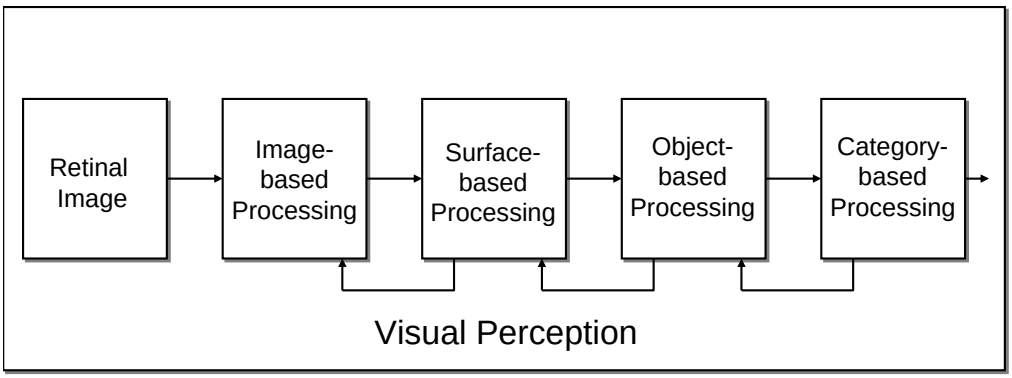


Figure 6.10: Four stages of visual processing: Visual processing can be divided into four main stages beyond the retinal image itself [Palmer 99].

In the first stage, some image processing operations take place such as detecting the edges, matching up the correspondence in the left and right image, and detecting the lines and their intersections. The surface-based stage is concerned with recovering the intrinsic properties of visible surfaces in the external world that might satisfy the information extracted from the image-based stage. Constructing such surfaces is a key to recover any 3D object in the scene, which is the goal of the object-based stage. This stage is responsible for recovering the 3D structure of the environmental objects. Finally, the category-based stage is concerned with recovering the functional properties of the objects. Within this stage, some classification rules are applied in order to identify these objects based on their visible properties, such as their shape, size, color, and location.

Our method for a feature extraction algorithm follows these four stages. A set of classification steps for each feature are set up based on this model, which can be summarized as follows:

- Prepare some required images (i.e. gradient image).
- Predict new surfaces depending on the region growing technique and a gradient image.
- Identify the feature.
- Classify the feature based on prior knowledge.

The algorithm has to work in indoor environments, and it has to follow these four stages in order to extract the features reliably. That means, the tasks of the algorithm are not

only the image segmentation and multiple feature detection, but the classification of these features as well. These tasks can not be resolved using only image information. Additional a priori knowledge has to be incorporated which is beyond visual information about the different features.

Within this section, we need to introduce the semantic definition of the type of these features we want to extract. So, we introduce the structure of these features in the next subsection, define their properties, and show how they can be extracted. Then we illustrate the algorithm by explaining each step, and showing its results. Finally, a procedure to detect some corners of a particular feature is presented in order to use them with the SLAM algorithm.

6.4.1 Analysis of Features

Definition

Features can be defined [Siegwart 04] as a recognizable structure of elements in the environment. They usually can be extracted from measurements and mathematically described, in order to enable more compact and robust descriptions of the environment.

For coping with the complexity of natural scenes we follow the model proposed by Palmer in order to analysis the appropriate features. The concept contains a close interaction between two levels, namely low-level features like lines, circles, or polygons, and high-level features such as doors, floor, roof, or tables.

We start with the feature analysis from top of this model down to its bottom, which represents the low-level description. The sensor model transfers many of the concepts of the features (3D object) to the image describing the expected appearance of the features. Figure 6.11 shows the types of knowledge with which a feature can be represented, which cover the following basic visual features: color, geometrical data, and texture.

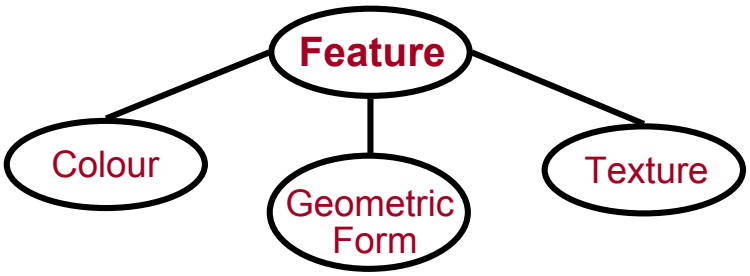


Figure 6.11: Structure of visual features

Color is a powerful cue in the extraction of features. That is because most features can be discriminated through their color properties. Thus, an extraction based on color, rather than just intensity, provides a better of discrimination between features. To detect regions of a specific color, we need an effective color model to accurately represent and identify colors under various illumination conditions. Among the applications involving colors, the RGB

color model is most widely used because of its easily implementable fashion. However, the inability of the RGB color model to separate the chromatic and luminance components of a color directly reduces the extraction performance. Therefore, it is useful to transform the RGB values to invariant property values such as in the HSV color model, which completely separates the luminance and chromatic components.

The geometric representation is an important property of the low-level visual features, which gives more knowledge about the features in terms of their shape, width, and height. In indoor environments, to be able to recognize the surrounding features automatically, one way is to extract a feature of a definite dimension and shape. For example, the extraction of a door requires the door frame and door dimension, which are the major characteristics of the door features. To find them, we can use their shape (e.g. the door frame has a rectangular form), or use their width and height (e.g. the door has a width one meter and a height of two meters). Figure 6.12 shows the description data needed to define the feature semantics if we use the geometrical information. This low-level geometry data can be used to get the high level description, which identifies a specific feature in the environment.

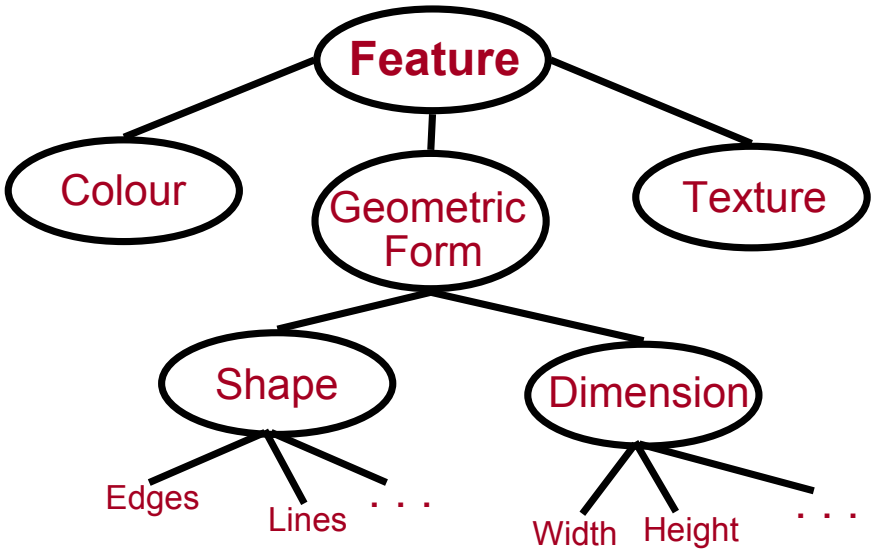


Figure 6.12: Extension of the structure of visual features in the case of geometric form

The texture on the other hand, is a very efficient way to describe the visual features. The texture can characterize the feature in order to discriminate it from other image contents. For example, doors in indoor environments have some texture information, which easily differentiates them from other features available in the environment. The texture properties of the door for example, can be described as door frame, or door knob.

Technically, to get a robust model of the human visual perception, some factors have to be taken into consideration. One of the most important factors is to find a suitable representation of each feature as prior knowledge, which can be helpful in the process of feature extraction. Indoor environments normally have clear line structures and large homogenous color surfaces. Both factors are helpful in classifying the natural features. So, all low-level

visual feature information is integrated to form a strong feature descriptor which can be used as prior knowledge in order to identify these features and classify them efficiently.

Types of Features

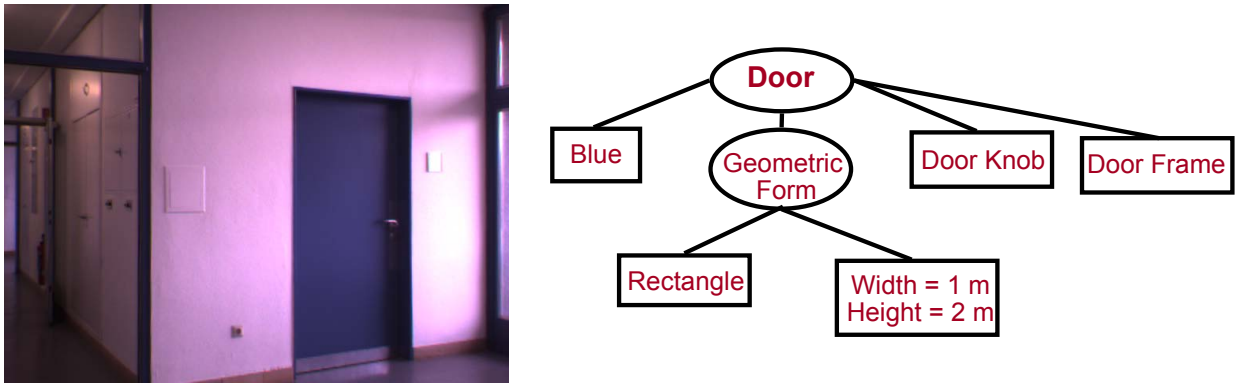
We follow the visual model by Palmer for describing the type of features used during the extraction process. In the image-based stage, the image structure is determined by finding some low-level features in terms of points, lines, circles, and ellipses. Attributes for lines, circles, and ellipses, for instance are orientation classifications, e.g. horizontal or vertical.

In the second stage, properties of visible surfaces are recovered which are induced by points, lines, circles, and ellipses, detecting all their direct neighbors. The surfaces are defined in terms of regions. Attributes for regions, for instance, are characterized in terms of their dimension, color and shape.

In the third stage, a 3D structure of these environmental features is built, also defining the texture of these features.

Finally, the features are identified based on this knowledge. In an office building, there are different types of features which can be extracted very efficiently. That is because, the office building is rich with structural elements that have different colors. Hence, the extraction of such features plays an important role in the creation of environmental models, helping the mobile robot during both map building and localization.

Figure 6.13(a) shows one type of features to be extracted, namely a door. Figure 6.13(b) presents the requirements needed to identify this feature as a door. The feature is classified as a door if its color is blue, if it has a rectangle shape, its dimension is of one meter width and two meter height, and if its texture has a door knob and door frame.

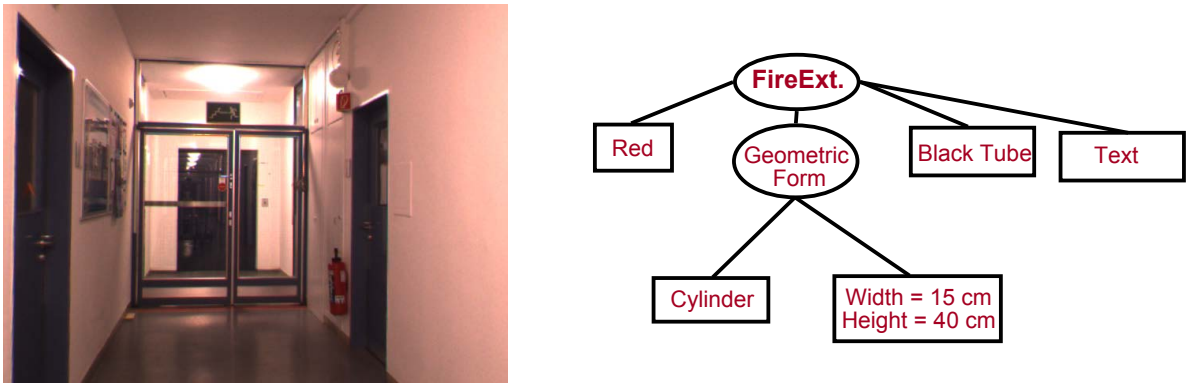


(a) (b)

Figure 6.13: Type of used features: (a) Doors in an office building can be extracted and (b) the door feature must satisfy the visual structure condition

Figure 6.14(a) presents an other type of feature to be extracted, namely a fire extinguisher. Figure 6.14(b) presents the requirements to identify this feature as fire extinguisher. The feature therefore is classified as a fire extinguisher if its color is red, if it has a cylinder form, its dimension is of $width = 15cm$ and $height = 40cm$, and if its texture has a black tube and some text are written onto it.

The information about the features like dimension and colors is used as a prior knowledge in the classification stage in order to get a more robust feature extraction process, and to discriminate these types of features from other indoor objects available in an image.



(a) (b)

Figure 6.14: Type of used features: (a) Fire extinguishers in an office building can be extracted and (b) the fire extinguisher feature must satisfy the visual structure condition

6.4.2 Feature Extraction Algorithm

The model for human visual perception introduced by Palmer [Palmer 99] proposes four stages in order to extract the relevant information found in the image. Our work follows these four stages. Figure 6.15(a) shows the steps of the classification process for each feature based on this model.

In the first stage, simple features like edges or corners are detected from the captured image. Subsequently, regions in the image corresponding to surfaces of real features are found in the second stage. After that, the detected surfaces are assigned to features in the third stage. Their role and relationship with other features within a scene are determined in the classification stage. This stage is presented in figure 6.15(b), showing that if no correspondence is found between right and left image then the object is not classified. In the next two subsections, these steps and their results are presented in more detail.

Feature Segmentation

The core of the developed algorithm combines the region-based and edge-based elements. So the scene analysis is carried out by using three image processing techniques: a *segmentation process* in the second stage of the model, a *feature detection* in the third stage of the model, and finally a *feature classification* in the fourth stage of the model.

As shown in figure 6.15(a), the first step is to prepare some pre-processing images in advance, which can speed up the execution time of the algorithm. Our approach for feature extraction is based on the observation that the features are characterized by their particular geometric form and their color. Therefore, we extract regions on the basis of geometric form and color information. On the one hand, we employ the HSV color space in order to verify the

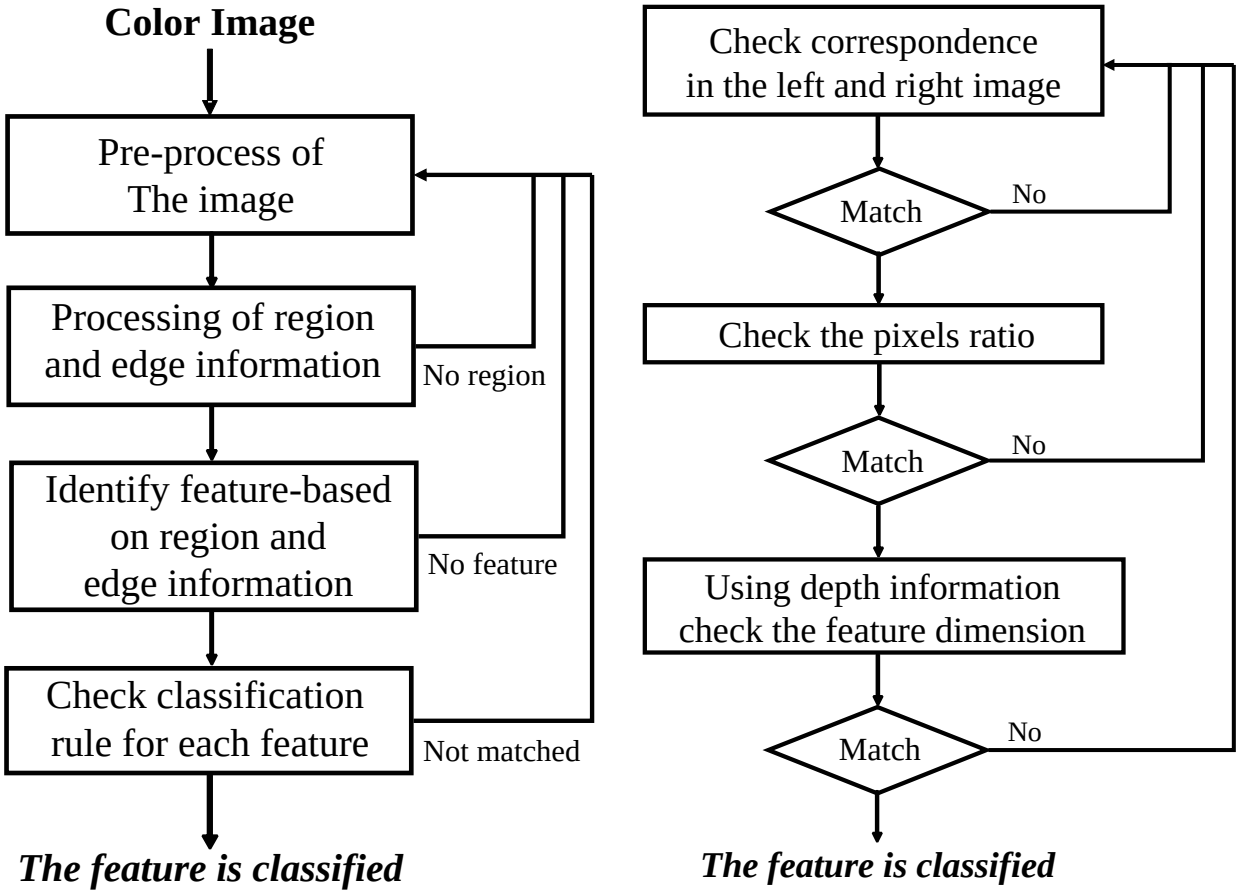


Figure 6.15: Concept of the feature extraction algorithm: (a) analysis of the features and (b) classification procedure

hypotheses for detected features by searching for visual feature properties inside the detected regions. On the other hand, we employ edge detection, which is used for two reasons. Firstly, the edge information is used to adapt the border to the contour of a feature. Secondly, it is used to identify the geometric model of the connected component in order to classify them as an accepted feature or not.

Therefore, two types of images are produced in this stage: The edge image using a canny-based method is produced. The edge detector we use is relatively quick and gives an accurate result. Figure 6.16(b) and figure 6.17(b) present the result of the edge detection of two types of features (displayed in figure 6.16(a) and 6.17(a)), showing as well the performance of this detector.

The second type of resulting image are HSV images, where each pixel in the input image of figure 6.16(a) and 6.17(a) is characterized by means of hue, saturation and value. Figure 6.16(c) and 6.17(c) show the result of the RGB to HSV transformation of the input image of figure 6.16(a) and 6.17(a), respectively.

In the second stage, a processing of region and edge information is performed, which is based on the region growing algorithm described in section 6.2. This method was chosen, because the segmentation process was developed taking into account some factors like pre-

(a) (b) (c)

Figure 6.16: Pre-processing of images: (a) the input image declaring the door as a feature of our interest, (b) the edge image using a canny edge detector and (c) the HSV image

(a) (b) (c)

Figure 6.17: Pre-processing of images: (a) the input image declaring the fire extinguisher as a feature of our interest, (b) the edge image using a canny edge detector and (c) the HSV image

cision and execution time in order to keep the algorithm applicable for real time landmark extraction. Therefore, a multi-level segmentation approach is used which considers edges as well as regions having similar colors. This approach avoids the lighting reflection effect, which produces arbitrary boundaries between adjacent regions. A gradient image, which is computed in the first stage, is compared with the segmented image in order to eliminate this effect.

As a result from the first stage, each pixel is modelled in HSV color space. The region of interest can be identified by the presence of a certain set of HSV values (H_r, S_r, V_r) as a feature-color reference map which is a prerequisite for the color segmentation. Since we are utilizing only the color information at this stage, the segmentation requires only the HSV components of the input image which was produced in the first stage. The output of the color segmentation, and hence step one of this stage, is a grey bitmap $O_1(x, y)$ described as follows:

$$O_1(x,y) = \begin{cases} 1, & if \ \ [h\,(x,y) = H_r] \cap [s\,(x,y) = S_r] \cap [v\,(x,y) = V_r] \\ 0, & otherwise \end{cases}$$

where x, y are the pixel position in the image. The h, s and v symbols are the corresponding values of the HSV color space of that pixel. The output pixel at point x, y is classified as

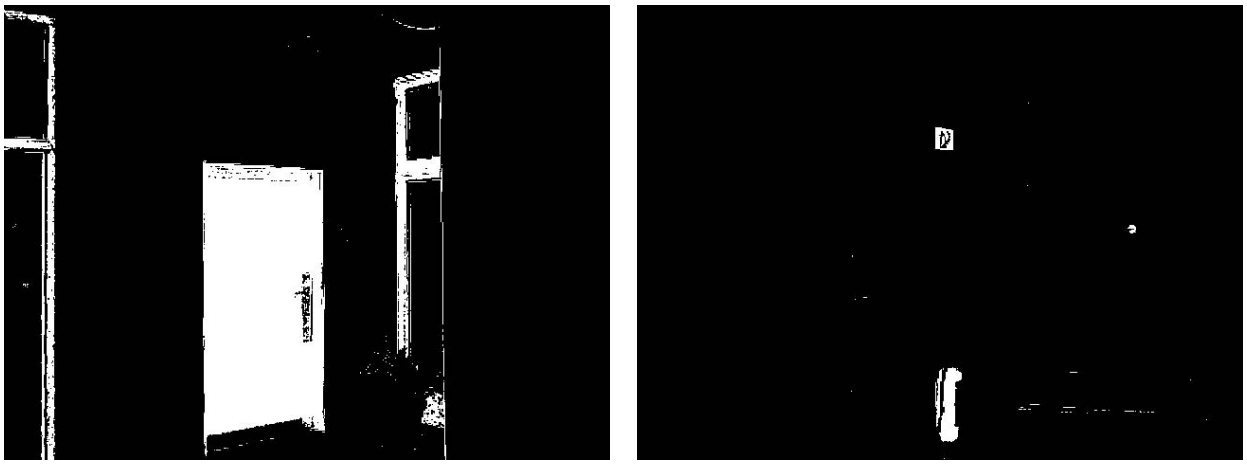
feature color and set to one if the condition at that point is satisfied. Otherwise, the pixel is classified as a non-feature color and set to zero.

In the second step of this stage, the region growing procedure described in section 6.2 is used in order to get a more refined detection result in a form of surfaces. The sequences of this step can be summarized as follows:

1. Search for pixels with a value of 1 which do not belong to any region in the output image O_1 in an order from the top left corner to the bottom right corner.
2. If a pixel $O_1(x, y)$ does not fit into any region, then a new region is created. Furthermore, we iteratively collect unused pixels that have the same value and are connective to $O_1(x, y)$. All of these pixels are labelled with the same region label.
3. If there are still unused pixels in the image, then step 2 is repeated. Otherwise, the segmented image is obtained.

The last step in this stage is to use the gradient image and to compare it with the segmented result in order to eliminate the lighting reflection effect.

To illustrate this stage, we perform color segmentation on the input image of two different places of indoor environment containing two different features, and the resulting bitmap can be seen in figure 6.18. The output value of one is printed in white, while the value of zero is printed in black. Figure 6.18(a) presents the color segmentation of the input image of figure 6.16(a), whereas figure 6.18(b) presents the color segmentation of the input image of figure 6.17(a).



(a) (b)

Figure 6.18: Color segmented images: (a) the color segmented image of the input image of figure 6.16(a) and (b) the color segmented image of the input image of figure 6.17(a)

The third stage of the feature extration algorithm (cf. figure 6.15(a)) is to identify features based on the surfaces found in the second stage and on the edge information produced in the first stage. This is done depending on some factors like the pixel filling ratio and the width-to-height ratio of each region. Another factor is whether each region in the output of

the gradient image has frame lines or not. This stage can be considered as a filtering pre-stage to the classification stage, in which regions with predefined geometric form or shape are selected as feature hypotheses.

As a result of Figure 6.18, a lot of surfaces with the same color are extracted. So, the feature detector runs on each segmented region and picks the feature with the highest probability based on the width-to-height region ratio and the filled pixels ratio of this feature. The shape of each connected region in figure 6.18 is evaluated. For example, the cylinder shape of the fire extinguisher can be approximated by using forms of ellipses and lines. Therefore, we compute the best fit cylinder for each connected region.

Furthermore, by using the gradient image, an adaptation of the edge of the feature is taken into consideration, finding the best region border. For our examples (e.g. for a door) we have found three lines representing the left, right and top edge of the door frame.

Feature Classification

The classification of the extracted feature as a certain structure in the environment is done in the final stage. In this stage, prior knowledge is introduced into the learning phase. Within this phase, the dimension of the feature and its color were stored for some features available in the environment to be used in order to extract them during the navigation. A sequence of this classification procedure is presented in figure 6.15(b).

Figure 6.19 shows two examples of a door and a fire extinguisher classifier available in the environment. The feature classifier detects each one correctly as a feature of our interest. The algorithm can recognize the feature if at least three quarters of the feature are appearing in both cameras. The red lines presented in figure 6.19(a) and 6.19(b) shows the boundary region of the feature frame presenting as well the adaptation of the edge detection which give us a more exact feature extraction. The yellow point in the middle of the door and fire extinguisher indicates that the door and fire extinguisher are classified correctly.

(a) (b)

Figure 6.19: Result of the feature extractor algorithm: (a) door classifier and (b) fire extinguisher classifier as the result of feature extractor

The stereo vision is used to perform the classification procedure, so that the color segmentation is performed on the left and right images. The properties of each region are its color, the bounding box, the centroid of this region, and the number of pixels being part of the region. Using this information together with the epipolar geometry, the correspondence problem can be solved very efficiently and effectively.

At the first step of this stage, for each detected feature from the third stage, the procedure checks the correspondence situation. If there is a correspondence of each feature in both images then the procedure goes forward to check another condition. If not, then these features are not classified as a feature of our interest. This phenomenon can be seen clearly in Figure 6.20(a), where the last door is not recognized by the right camera.

At the second step of this stage, if the pixel ratio of the detected feature obtained from the third stage lies below a given threshold, this feature will then be rejected and will not be classified as a feature of our interest.

At the third step of this stage, the dimension of each detected feature is checked using depth information. By using a stereo camera, the depth information can be obtained in order to get the geometric properties of each detected feature. This information is used in this stage to define the category class of this feature. Figure 6.20(b) presents this phenomena, where the fire extinguisher is correctly classified, whereas the red bin is not classified because of its different geometry properties.

(a) (b)

Figure 6.20: Interesting classification cases in the result of feature extractor algorithm: (a) a mismatch of the feature correspondence of the third door left in both cameras and (b) the same color as a valid feature, but different geometry and dimension

Experimental Results and Discussion

In general, the feature extraction algorithm we presented produces a reliable and sensible autonomous extraction of features in an office building. Two types of features were correctly extracted, namely doors and fire extinguishers. The extraction process is based on the color, geometric form, and texture of these features.

The processing time of the approach is suitable for real time applications, and it was applied within the exploration strategy presented in chapter 4. Within a few seconds, the relevant features were extracted and classified as being some known features in the environment. The algorithm produces a stable extraction process under different situations in the environment during both the day and the night. The usage of edge data was shown to improve the lighting fluctuation problem without losing the advantage of the real time applicability.

Feature extraction was performed at different positions in an office building, showing the stability of this algorithm. Figure 6.21 presents that a number of the results obtained in different situations within a day or at the night and in different places. These results show that different features can be extracted in one picture as shown in figure 6.21(e), and many features (up to 7 features in this example) can be extracted in one picture as shown in figure 6.21(f).

However, the algorithm needs to be more robust against other structure of the indoor environment to classify them as well, for example the floor or the ceiling which do have not a geometric form. The algorithm also needs to be generalized to be applied on other environments such as outdoor environments. Another restriction of our algorithm depends on stereo camera itself. The same feature must be detected in both cameras in order to recognize and to classify the feature.

6.4.3 Corner Detector Algorithm

In order to use the results of the last subsection as a reference for the SLAM algorithm, we detect some point feature to be observed by the SLAM algorithm. The most robust point feature is the corner of the features, which can easily be labelled, and can be observed quite well given a small view of the camera. Furthermore, detecting corners can be more computationally efficient than detecting other features such as lines or some texture. Furthermore, the correspondence process can quickly be done in efficient way.

There are several approaches for detecting corners which can be divided into two groups:

- ***Direct Detectors*** contain algorithms that work directly on a grey-level image without segmenting the image in advance. A matrix of cornerness of the points is computed based on the gradient magnitude and on the rate of change in gradient direction. Then points with a value over a given threshold are selected as corners.
- ***Boundary-based detectors*** use algorithms that extract the boundaries of the objects first and analyse their shape afterwards. In these approaches, the edges are extracted first, then it is searched for corners, which are selected based on a curvature calculation. The points with maximum curvature are selected as corners.

We have evaluated two algorithms that work on the grey level image and detect corners depending on the brightness of the pixels, either by minimizing regions of similar brightness [Smith 97] or by measuring the variance of the directions of the gradient of brightness [Sojka 03]. These two methods produce corners without taking into account the color of the

(a) (b)

(c) (d)

(e) (f)

Figure 6.21: Results of feature extractor algorithm with different conditions: (a), (b), (c) and (d) The extraction process during the day at different positions, (e) and (f) the extraction process during the night with different types of features in one picture.

regions. As we are interested in detecting a corner of a specific feature (e.g. a door), the corner detector is applied only on our region of interest to speed up the processing time.

We have tested both methods on different situations of the features, and we have found that the gradient-based method [Sojka 03] is more stable and robust than the SUSAN method [Smith 97], satisfying the following requirements:

- The detection of corners is highly reliable.
- The location errors are small.
- The sensitivity to noise is low.

Thus, we have selected the gradient-based method of [Sojka 03] to implement the corner detector in our approach, and we will call it the *Sojka corner detector*.

The main ideas of the Sojka corner detector can be summarized as follows:

- The algorithm determines the corner response function that combines the angle and the contrast of the corner. The function is designed in such a way that it exhibits its local maxima at corner points.
- In figure 6.22, the magnitude of the gradient of brightness of the areas A, B and C are non-zero values. The algorithm is designed in such a way that it determines which areas are relevant for deciding whether or not Q is a corner. It is done by introducing the probability $P_{sg}(X)$ of the event that an arbitrary point X in the neighbourhood $\Omega(Q)$ belongs to the approximation of the straight segment containing Q of the isoline of brightness. The value of $P_{sg}(X)$ is computed from the values of the function of brightness and its gradient by applying the Bayesian estimations. In figure 6.22, the values of $P_{sg}(X)$ are high at the points that form the area A, whereas at all other points, the values of $P_{sg}(X)$ are low.
- In figure 6.22, for determining the angle α of the corner at Q, obviously only area A is relevant, the area B is less substantial, and the area C is almost irrelevant. The algorithm takes this fact into account and selectively examines the whole neighbourhood of Q. The expected precision of the angle measurement is estimated, and a point Q can only be accepted as a corner if the difference is significantly greater than the estimated precision of the angle measurement.
- A point Q can only be accepted as a corner if its apparency is greater than a predefined threshold. The apparency quantity is computed based on the size of the area that is relevant for deciding whether or not Q is a corner and on the magnitude of the gradient of the brightness in this area.

As a result of including the Sojka corner detector into our framework, we get a point features that can be used in the SLAM algorithm. Since we already know the position of features of interest, it is not effective to try to detect point features on a whole image. A more efficient way is just to investigate a certain area around the feature of our interest, which is known

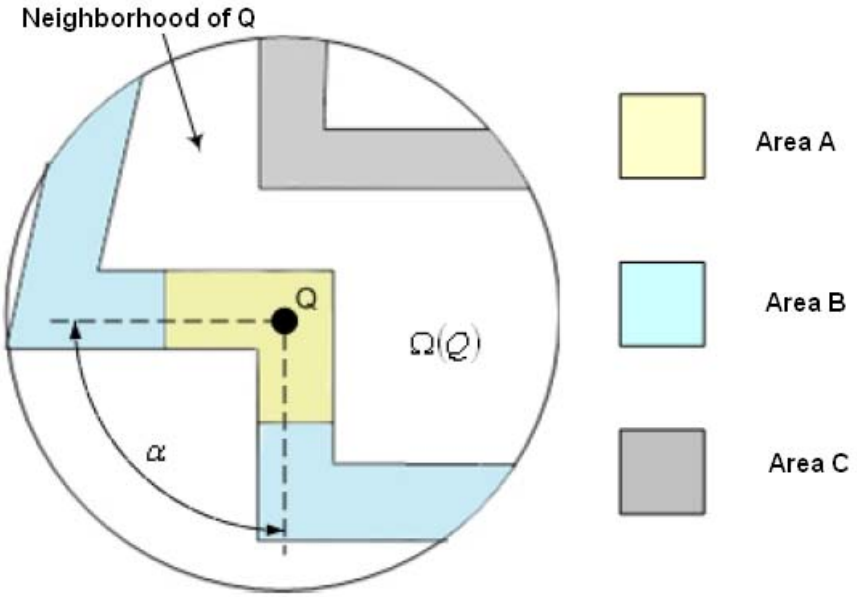


Figure 6.22: A neighborhood of pixel Q. In Area A, B and C, the magnitude of gradient of brightness are non-zero values. Area A is relevant for determining the angle of corner α at pixel Q. Areas B and C are all irrelevant for determining the angle of corner α at pixel Q.

(a) (b)

Figure 6.23: Point feature detection: (a) result of the Sojka corner detector and (b) result of the feature extractor algorithm

as region of interest. This selection save a considerable amount of processing time and give more precise corner locations. By using stereo camera images, one more prerequisite can be checked, namely the correspondence. If there is no correspondence between the left and right images concerning any detected corner, then this corner is rejected, which gives us more stable results. In figure 6.23, a result of the corner detector algorithm is shown. The corner algorithm detects the upper corners of the door efficiently for further use in the SLAM process. The blue rectangles indicate the detected corners.

6.5 Range Image Processing

6.5.1 Range Images

Intensity images are of limited use in terms of an estimation of surfaces. Pixel values are only related to the surface geometry indirectly, whereas range images encode the position of a surface directly. Therefore, the shape can be computed reasonably easy. Range images are a special class of digital images. Each pixel of a range image expresses the distance between a known reference frame and a visible point in the scene. Therefore, a range image reproduces the 3D structure of a scene.

Range images can be represented in two basic forms. One is a list of 3D coordinates in a given reference frame that is known as cloud of points, for which no specific order is required. The other is a matrix of depth values of points along the directions of the x,y image axes, which makes spatial organization explicit.

Range images are acquired with range sensors. In computer vision normally optical range sensors are used. We can distinguish between active and passive range sensors. Active range sensors project energy (e.g. light) on the scene and detect its position to measure or exploit the effect of controlled changes of some sensor parameters. On the other hand, passive range sensors rely on intensity images to reconstruct the depth.

To acquire range images, we use a 3D laser scanner (RoSi) which is based on a commercial 2D scanner with a movable origin. Figure 6.24(a) presents the RoSi scanner II. The scan is achieved while rotating the RoSi sensor around its z-axis by means of a DC-motor.

(a) (b) (c)

Figure 6.24: Active range sensor: (a) 3D laser scanner RoSi II, and (b) its corresponding coordinate system, and (c) the digital camera of the RoSi scanner

Spatial Data Acquisition

We use a Sick LMS 200 scanner in RoSi II that has the following properties:

- The scanning cone is set to 100°. It is possible to change it to 180° which results in a complete hemisphere.

- The angular resolution is set to 0.25°. That means, we get 401 measurements for each scan line. It is also possible to configure it to 1° or $o.5°$.
- The depth resolution lies within a $\pm 2cm$ boundary, and the frequency of the scan is set to 20Hz.

By the rotation of the scanner, the overall system receives all the scan data which is lying in the direction of the cone with an opening angle of 100°. At a firmly fixed rotation speed of 30 per second, the scanner measures a half resolution in 6 seconds, and thereby receives a complete depth image. At this rotation speed, the angular resolution amounts to approximately 1.5° per scan. A depth image then consists of $6 \times 20 \times 400 = 48000$ measurement points.

As shown in figure 6.24(b), the scanner coordinate system is oriented in such a manner that the z-axis corresponds to the rotation axle of the scanner, the x-axis lies in the horizontal scan with rotation angle equal to 0, and the y-axis accordingly shows orthogonally upwards.

Then, a horizontal scan point without rotation would have the coordinates:

$$x_i = \cos(\alpha_i) \,.\, r_i$$
$$y_i = 0$$
$$z_i = \sin(\alpha_i) \,.\, r_i$$

with α_i as scanning angle and r_i as corresponding measured distance. Now, if we take the rotation of the scanner around the rotation axle into account with β as the rotation angle after the scan, the coordinates of scan point i amount to:

$$x_i = \cos(\beta)\cos(\alpha_i) \,.\, r_i$$
$$y_i = \sin(\beta)\cos(\alpha_i) \,.\, r_i$$
$$z_i = \sin(\alpha_i) \,.\, r_i$$

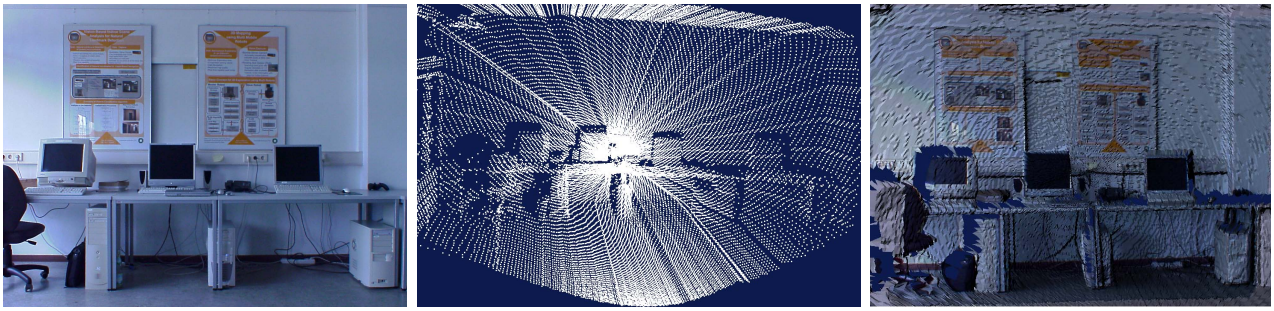
Furthermore, the actual scan time of $t_s = 20ms$ is the time which the laser needs in order to record a single line scan, then the rotation of the scanner within this time must be considered as well:

$$x_i = \cos(\beta - (400 - i)\, RG/\,(400 t_s))\cos(\alpha_i) \,.\, r_i$$
$$y_i = \sin(\beta - (400 - i)\, RG/\,(400 t_s))\cos(\alpha_i) \,.\, r_i$$
$$z_i = \sin(\alpha_i) \,.\, r_i$$

with RG as the actual rotation speed in (rad/s).

Sample of Range Image Representation

Figure 6.25 shows a sample of a range image with 8m maximum range and the corresponding original image captured by the digital camera above the RoSi sensor that can be seen in figure 6.24(c). Figure 6.25(b) shows the resulting point cloud of the scene, whereas figure 6.25(c) shows the same data as triangulated surface model. Walther et al. presented [Walther 06] several point clouds acquired with different configurations and of different situations.



(a) (b) (c)

Figure 6.25: Sample of range image: (a) The scene captured by the digital camera equipped with RoSi System, (b) its corresponding point clouds range image, and (c) its corresponding triangulated point cloud

6.5.2 Extraction of Geometrical Features from Range Images

The goal for the range image processing by the slave equipped with a RoSi system is the extraction of geometric features relevant to the features already extracted by the master robot equipped with a stereo camera.

The RoSi system is complemented with a digital camera (see figure 6.24(c)). This camera projects the observed 3D scene onto a 2D image plane. So the resulting digitized data does not explicitly contain the depth information. To perform the goal mentioned above, it is highly desirable to obtain 3D information about the geometry of the scene. Thus, a RoSi system is used with a range measuring technique such as the laser scanner. This sensor is developed in such a way that it takes a pair of range and intensity images of a given scene as input and then outputs a 3D reconstruction of that scene composed of a geometric description of known sizes and locations that represents the features of interest in the scene.

Even though good quality range images are available, the intensity data is also used to achieve a better scene description such as the color of the feature. To register intensity and range data, it is necessary that the pixel location of a point in the scene corresponds in the intensity and range images. The texture integration process is explained in the next subsection.

A requirement for the extraction of geometric features for the position estimation of the slave robot is that the localization of the regions, especially those corresponding to the features of interest, must be as precise as possible, and that the classification of these regions must be correct. Based on these criteria, the process incorporated in this work is divided into three steps:

1. Detecting a region of interest based on prior knowledge.
2. Integrating the texture of these regions into the range image data.
3. Classification process.

Algorithm 6.3 presents the basic idea of this process. A pair of range and intensity images of a given scene are acquired from the RoSi sensor. They are used as input to algorithm 6.3.

Input: Img, P /*Img is a Color image of the scene*/
/*P is a 3D point clouds of the scene*/
Output: F /*List of all detected features*/
1: Show(P) /*Read and represent the point clouds*/
2: RegionGrowing(Img, R) /*Get a region list that satisfies the color condition*/
3: **for** *each region in* R **do**
4: **if** $w_{2D}(R_i) > h_{2D}(R_i)$ **then**
5: Delete R_i /*If the width of the region is greater than its height then reject it*/
6: **else**
7: Fuse($R, Color$) /*Fusing a texture of the found regions only*/
8: Classifier(R_i, F) /*Run the classifier, and add the region to the feature list if it satisfies the condition*/
9: **end if**
10: **end for**
11: **Return** F

Algorithm 6.3: The algorithm to extract geometrical features from range images

As a result, features of interest are found and classified. The basic idea of the classification procedure is presented in algorithm 6.4, whose idea is to fit pre-defined feature models to the obtained region, and to then make a decision to either accept or reject the presence of such features.

The method used to detect regions of interest is based on the region growing techniques described in section 6.2. The basic principle of this method is to extract regions on the base of geometric form and color information. The method is presented in more detail in subsection 6.2.2.

As a result of the region growing method, some regions that are potential features of interest are found. A quick filter is needed here in order to reject inappropriate regions such as small regions or regions whose width is greater than its height which does not satisfy our requirements. Within this step, the algorithm does not need the 3D information of these regions to reject them.

The next step is to integrate the texture of these regions into the range image data which will be used in the classification step. As a result, only the extracted regions are overlaid with their texture.

The classification of the detected regions as a specific structure in the environment takes place in the last step. Algorithm 6.4 presents the sequential steps of this classification based on a priori knowledge of the dimension of the expected feature and its color.

The algorithm can accept the region as a feature of interest if the 3D dimension of any region is corresponding to the prior knowledge we have. The 3D information is obtained from the range image data, which gives us the dimension of the region accurately if the robot sees it from the front view. Otherwise, if the robot sees the surfaces from a side view, then a view angle is calculated in order to take this into account. A virtual rotation is done on this surface to calculate the width and height of this surface correctly.

Feature detection results are shown in figures 6.26 and 6.27. It can be seen that both doors and fire extinguishers are detected well. The features are classified correctly if their

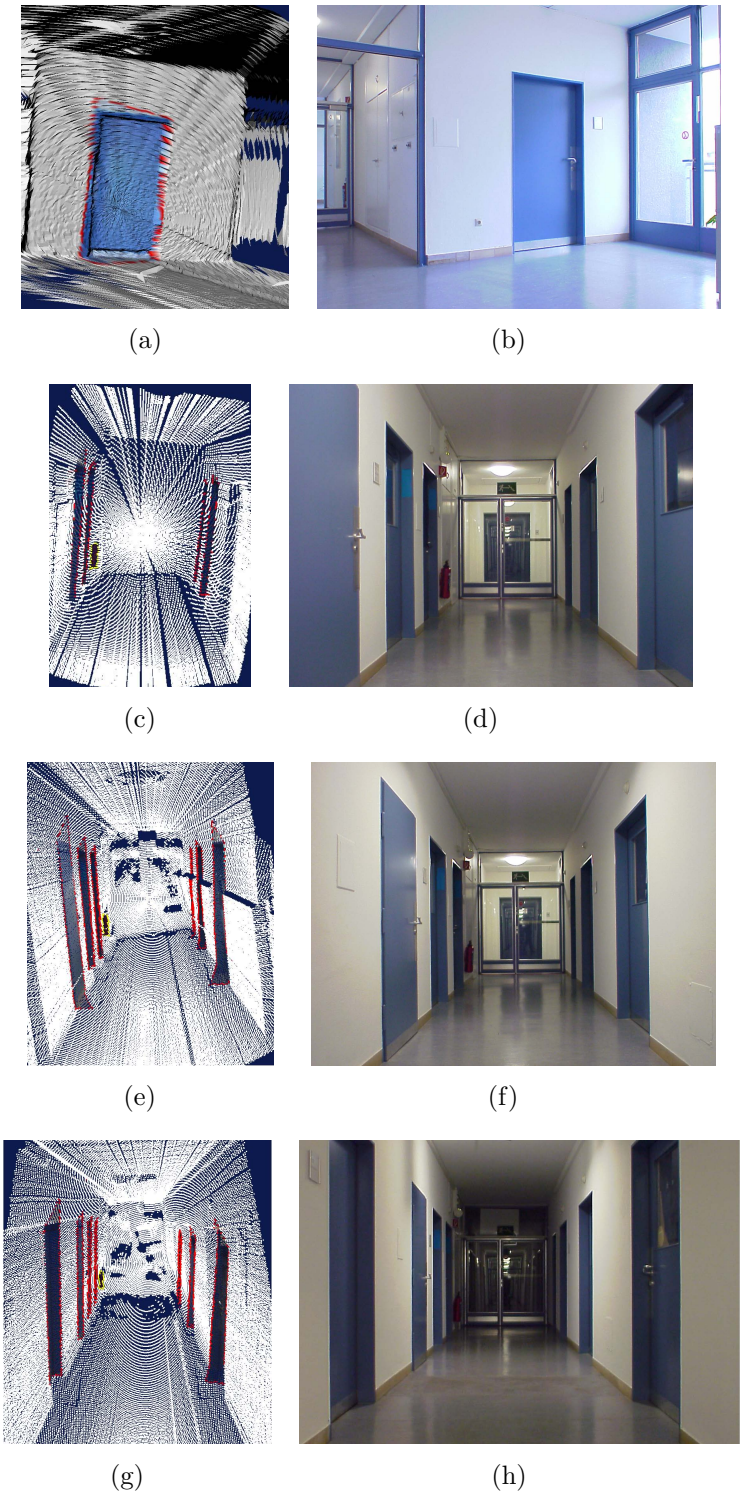
Input: R_i /*The region R_i to be classified*/
Output: F /*List of all detected features*/
read τ_f /*Read corresponding threshold value of each feature */
read h_f, w_f /*Read corresponding width and height of each feature*/
/*If ratio of this region is less than the threshold, then this feature is seen from the side view*/
if $Ratio(R_i) < \tau_f$ **then**
  Find θ_{R_i} /*Calculate the view angle*/
  $R_i = R_i(\theta_{R_i})$ /*Rotate the feature by θ_{R_i} to be seen from the front view */
end if
Get $D(R_i)$ /*Compute the distance to the region R_i*/
Get $w_{3D}(R_i), h_{3D}(R_i)$ /*Calculate the 3D dimension of the region R_i*/
if $w_{3D}(R_i) \neq w_f \| h_{3D}(R_i) \neq h_f$ **then**
  Delete (R_i) /*If the condition is not satisfied, then reject the region R_i*/
  Return 0
end if
$F \leftarrow R_i$ /*Add this region to the list of features*/
Return F

Algorithm 6.4: The feature classifier algorithm

distance from the robot falls within 2 to 8 meters and if the feature itself appears in the image captured by the camera. The red lines presented in figures 6.26 and 6.27 represent the boundary region of the feature frame, indicating that the feature is correctly classified as a door, whereas the yellow lines show that the feature is correctly classified as a fire extinguisher. The texture of only the region of the feature of interest is fused into the range image data.

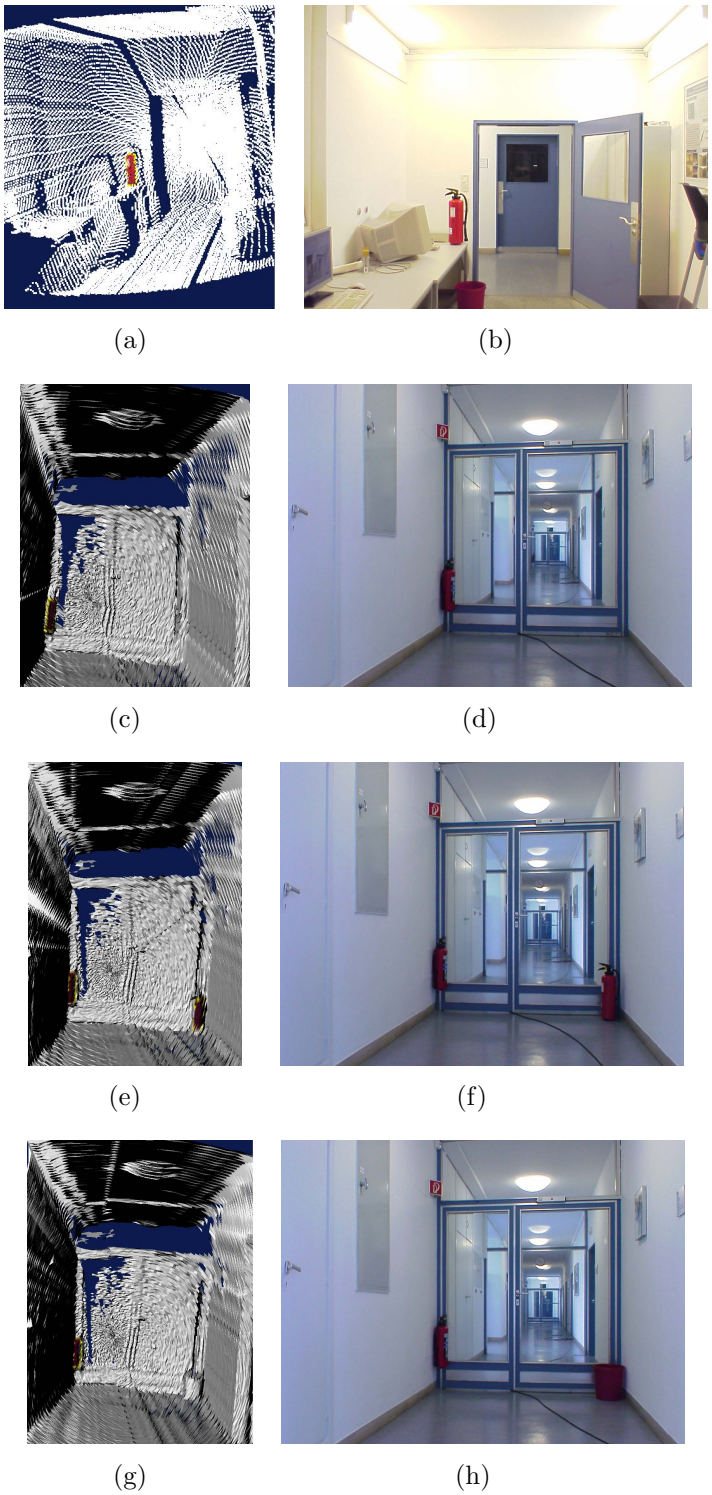
In figure 6.26, the results are obtained at different environmental places. Figure 6.26(a) shows a triangulated point cloud of a feature correctly classified as a door. Figure 6.26(c) shows a range image point cloud with four doors and one fire extinguisher that are correctly classified. In this figure, the first door to the right and to the left are not classified because they are out of the view from the camera. Figure 6.26(e) and 6.26(g) show how stable the algorithm is, as all the doors are extracted correctly with a very precise location estimation. The algorithm extracted up to seven features in one capture process if all the conditions are satisfied.

Figure 6.27 shows the results of the range image processing obtained on another feature, namely a fire extinguisher. The picture are taken at different situations. In figures 6.27(a) and 6.27(c), a feature is detected efficiently and is classified correctly as a fire extinguisher. Figures 6.27(e) and 6.27(g) present the results of the classification procedure if we have two features of the same type or if there is a another feature with the same color but with different dimensions. In both figures, the features are correctly classified as a fire extinguisher.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.26: Results of range image processing at different environmental positions: (a) detection of a door, (b) its original image, (c) and (e) and (g) detection of different features at the same time, (d) and (f) and (h) their original scene, respectively.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.27: Results of range image processing at different environmental positions: (a) and (c) detection of a fire extinguisher, (b) and (d) corresponding original images, (e) and (g) the result of the classification procedure with (f) and (h) original images.

6.5.3 3D Surface Reconstruction

The scene reconstruction from the point clouds is done by adding neighbouring points to a net of triangle elements which can then be represented as a pure mesh net. Thereafter, the graphic data capture by the camera can be added for texture integration.

The mesh representation depends on the presence of a triangulation. That can easily be computed from the implicit order of the points on the scan rows, without falling back into the time-intensive computation of the existing standard methods of the triangulation for general point clouds.

Figure 6.28 presents the visualization of the triangulation algorithm. As can be seen from this figure, it is necessary to observe the following changes: The sequences of the point of triangle must be exchanged to the middle point of each scan raw (e.g. the point k), otherwise the normal triangle points to the wrong direction. That means, the triangles in the old pattern are formed by connecting the points i, i+1 of scan row L ($P_{i,L}, P_{i+1,L}$ respectively) with the point i of scan row L+1 ($P_{i,L+1}$), whereas the triangles in the latter pattern are formed by connecting the points j, j+1 of scan row L ($P_{j,L}, P_{j+1,L}$ respectively) with the point j of the scan row L+1 ($P_{j,L+1}$). With the latter pattern, the sequence of the triangle must therefore be changed to $P_{j,L}$, $P_{j+1,L}$ and $P_{j,L+1}$ so that the sense of direction matches that of the other triangles. It is also necessary to consider the triangles which do not belong to a reflected scene. They can be avoided by using a variable range threshold.

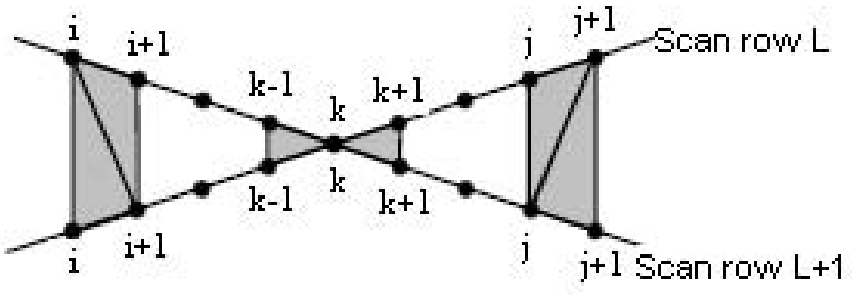


Figure 6.28: Visualization of the triangulation algorithm

In order to integrate the texture, the digital camera attached above the scanner is used. The camera supplies color images which can be used to assign appropriate color values to the points in the display space to receive the texture of each individual triangle. Due to the different capturing positions between the scanner and the camera, false allocations are possible. Therefore, a calibration between the 3D coordinate system of the scanner sensor and the 2D coordinate system of the digital camera is necessary. The calibration process is explained and demonstrated in [Walther 06]. As a result of this calibration, the texture is integrated correctly on every corresponding triangle by assigning the corresponding image pixel to every point belonging to the triangle.

Figure 6.25(c) shows a result of the triangulation algorithm, where the output image is a visualization of the point clouds in figure 6.25(b) as surface model. This algorithm was developed by Walther and is demonstrated [Walther 06]. Within our work, we use this algorithm in order to demonstrate our results captured by the slaves robots.

6.5.4 Combining Multiple Range Images

One range image represents only a part of the 3D environment. To continually build a complete 3D map of the environment, the robot should move around in order to scan from different viewpoints. Within our work, the task of a master is to send their team of n slaves to certain places in order to get a 3D map of those places. The slaves send the range image data back to their master in order to fuse all of it in one 3D model.

Fusing representations from different viewpoints produces a description of the environment with more information than any of the individual descriptions. To do that, the master follows these steps:

1. Receive each view point of different places knowing both the position and orientation of the corresponding slave.
2. Estimate the 3D rigid transformation.
3. Fuse each view point into the global map.

At the first step, the slave estimates its position and orientation by extracting some known features processed by the master. Then, it starts scanning its place. A 3D map is built and is sent to the master in order to merge it into the global map.

At the second stage, the 3D transformation is represented by the 4×4 homogenous transformation matrix T,

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix},$$

where the matrix R is a 3×3 rotation matrix which specifies the orientation of the slaves, and t is a 3×3 translation vector.

By using the transformation matrix T, the single-view range image data can be transformed from its local coordinate system to a global map coordinate system. A result of this step is that the new single view is aligned with the data sets acquired from other slaves at different positions.

Several experiments have been conducted in our laboratory concerning the combination procedure. In the first experiment, we used the data captured by the slave at different places in the corridor. The slave performs four laser scans. Parts of the resulting can be seen in figure 6.29. From top to bottom, figure 6.29(a) presents the range image data obtained at the first location, figure 6.29(b) presents the fusion of the first scan with the result at the second position of scanning, and so on. Table 6.1 shows the scanning position of the slave at each time.

In the second experiment, the merging process is carried out on a section of our laboratory that has two doors. Figure 6.30(a) and 6.30(b) shows the range images of scanning the left and right doors, respectively. In figure 6.30(c), a merging process was performed on this data to get a complete model of the laboratory.

Scan No.	x_{mm}	y_{mm}	θ_{rad}	
1	0	0	0	first scanning position at a corridor of our building
2	5966.8	-13.25	-0.013	second scanning position at a corridor of our building
3	10023.0	139.51	0.0536	third scanning position at a corridor of our building
4	14230.1	-115.9	0.0414	fourth scanning position at a corridor of our building
5	8623.2	1118.5	1.5613	scanning position at the first door of our lab
6	12258.0	1312.4	1.6553	scanning position at the second door of our lab

Table 6.1: Scanning position of the slave at different places in the environment

It can be concluded from these results that the merging process is successful. However, the process is performed without reducing the overlapping scan portions of each range images. A scan matching algorithm is needed in order to register each scan correctly without any overlapping. Although the registration process is an important issue, especially if an accurate 3D model of the indoor environment is required, it is out of the scope of this thesis and thus it is not addressed here.

6.6 Conclusion

This chapter was concerned with the feature extraction techniques used within the current work. It started with the discussion of color image segmentation by introducing different color spaces in different methods of segmentation.

The second section introduced the basic definition of the region growing segmentation method, in particular demonstrating the sequential steps of the algorithm. This method is based on the HSV color model to identify the regions of interest. In order to get a robust segmentation result, the conditions stated in the definition should be satisfied. The algorithm consists of three steps, starting with determining the initial seed regions depending on prior knowledge of the feature color. Then the region is grown, beginning from these seed regions. Finally, a post-processing is done in order to reject or merge insignificant regions depending on a priori knowledge of the feature dimension.

The third section introduced the basic idea of another segmentation method, namely the mean shift segmentation method. This method is introduced in order to avoid the requirement of a priori knowledge. Mean shift segmentation is a non-parametric density estimation technique, which performs the segmentation process autonomously without any prior knowledge of the feature. Its robustness is good, however it suffers from the high processing time required to do the segmentation.

In the fourth section, the concept of the feature extraction algorithm is presented. This algorithm uses the region growing segmentation method in order to predict new surfaces. A classification procedure is implemented based on a prior knowledge. The prior knowledge of a feature is represented in form of its color, its geometric form and its texture. The results

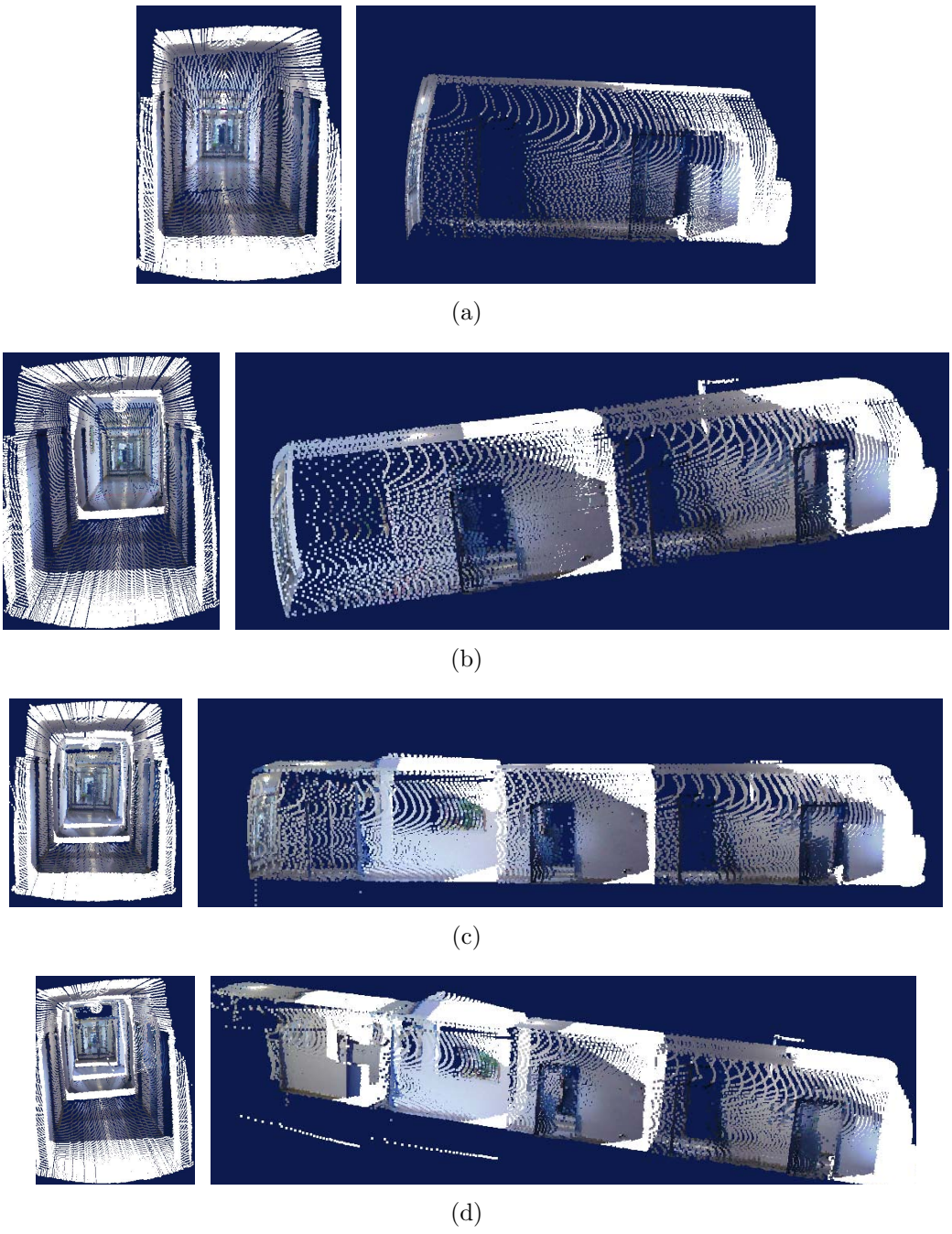
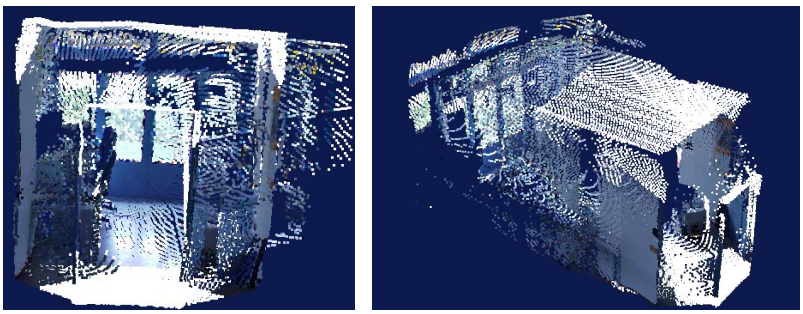


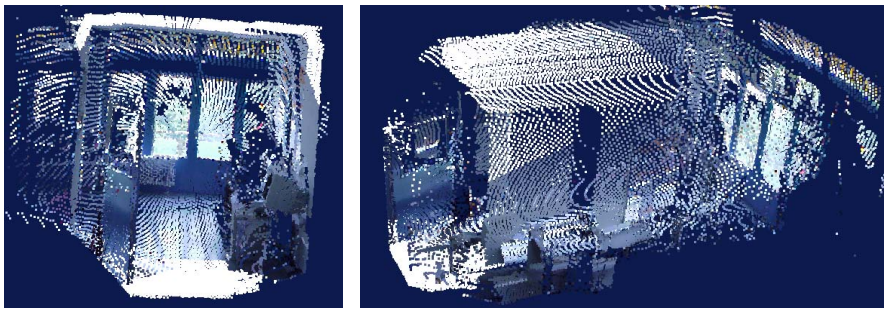
Figure 6.29: Results of range image merging at different scanning positions in the corridor at our lab: (a) the first scanning result by the slave, (b) merging two range images, (c) merging three range images, and (d) merging four range images.

of this algorithm are also presented showing reliability. Two type of features were tested within this algorithm, namely doors and fire extinguishers. At the end of this section, Sojka's corner detector was presented, whose results are used in the SLAM algorithm.

The fifth section was concerned with range image processing. It started with a definition of range images. The extraction of geometrical features from the range images was introduced as well. The result of this algorithm shows how stable the algorithm is. The RoSi system was demonstrated within this section, which is used to perform the range image processing.



(a)



(b)

(c)

Figure 6.30: Merging of two scans carried out at a section of our laboratory that has two doors: (a) the scanning result at the first door of our laboratory, (b) the scanning result at the second door of our laboratory, and (c) the result of merging these two into a final range image.

The idea of how a 3D surface is reconstructed was discussed. Finally, the merging procedure of multiple range images captured by different slaves was defined.

Chapter 7

Evaluation

The system performance of the presented work was evaluated with regard to the motivation and initial requirements as stated in chapter 1. The primary motivation for the work presented here was to develop a robust master-slave system for exploring indoor environments, considering the way how robots are coordinated over the environment and the time of the exploration.

The evaluation criteria which can be consulted are then on the one hand the exploration strategy concerning the time of exploration, the degree of cooperation and the coordination efficiency. On the other hand, the quality of the algorithm to extract a natural landmark has to be evaluated concerning the time of the feature extraction process and the stability of the feature extraction at different places in the indoor environment. The reason of considering the feature extraction algorithm into the evaluation step is that it is a large part of the system which uses two different sensors, and as such it may considerably affect the overall system performance due to the time consuming nature of this algorithm. Within this chapter, it is to be considered in particular whether the soundness of the theses underlying the current work (as presented in section 1.3) could be convincingly shown.

7.1 Performance Evaluation of the Master-Slave System

To evaluate the performance of the developed master-slave system and to get an objective view of its merits, limitations and prospects, we implemented a simulation package that treats the work of the masters and slaves separately, because of the different nature of their tasks. We firstly evaluate the performance of the master, and then the performance of the slave. Both cases have different performance criteria, however the main evaluation criterion is the overall time of execution a task.

7.1.1 Master Case

Three strategies were implemented in order to evaluate the performance of the masters. The first strategy performs the exploration using the frontier mechanism and prospect values

presented in chapter 4. The second strategy extends the first one, trying to avoid the limitation of the first strategy. In the third strategy, a dynamical technique was used in order to let the robots cooperate more efficiently and to finish the task in the same time.

The evaluation criteria for comparing the three strategies mentioned before are the cooperation degree, the exploration time, the speed of each robot and the area of the environment.

The least cooperation took place with the first strategy in which the robots are cooperating only at the beginning of the exploration in order to choose different directions for exploration. This limitation was improved with the sub-areas strategy in which the robots are cooperating each time when they finish exploring the current sub-area and start exploring a new one. Within the third strategy, the robots are strongly cooperating during the exploration process, in that they cooperate as soon as an event occurs such as the detection of a feature or the completion of a sub-area.

Figure 7.1 presents the second evaluation criteria, i.e. the performance time for the exploration task for all three strategies. It can be noticed that the dynamic strategy results in the best exploration time. Furthermore, it can also be noticed that the use of multiple robots results in a better exploration time than the use of a single robot.

As shown in figure 7.1, the evaluation test was performed on two different environment areas. It turned out that as the area of the environment becomes larger, more time is required in order to do the exploration and thus it is more efficient to use a multiple robot system.

As a result of evaluating the degree of cooperation, the exploration time and the area of the environment, we found that the dynamic strategy has a better exploration time, the best cooperation degree, and the best partitioning of the whole area, because the robots are cooperating all the time. However, this strategy can be applied only to certain applications with partially known environment. On the other hand, the sub-areas strategy is more suitable to our work, where the partitioning of the whole area into sub-areas improves the computational and storage requirements considerably.

From Figure 7.1, it can be noticed that the time of the exploration in the case of the sub-areas strategy is the worst time of exploration amongst the three strategies. This is the case only if the robots have the same speed, as can be seen in figure 7.2. In the case of two robots with different speed, the sub-areas strategy is advantageous to be with a better performance than the other strategies, since the degree of cooperation was increased in this case and the masters were achieved their tasks in an efficient way. From figure 7.2, It can be noticed that the exploration time of all strategies is considerably improved compared to the case of only one robot by more than one half of its exploration time. When we compare both figures 7.1 and 7.2, we can find that, the sub-areas strategies improves the time of the exploration, whereas the time of the exploration in the case of the frontier strategy is the worst. Thus, the result from figure 7.2 shows that the sub-areas strategy has a better performance for multi-robot systems in general.

7.1.2 Slave Case

Two strategies were implemented in order to evaluate the performance of the slaves. The first strategy has a centralized form in which each slave receives a list of tasks to be carried

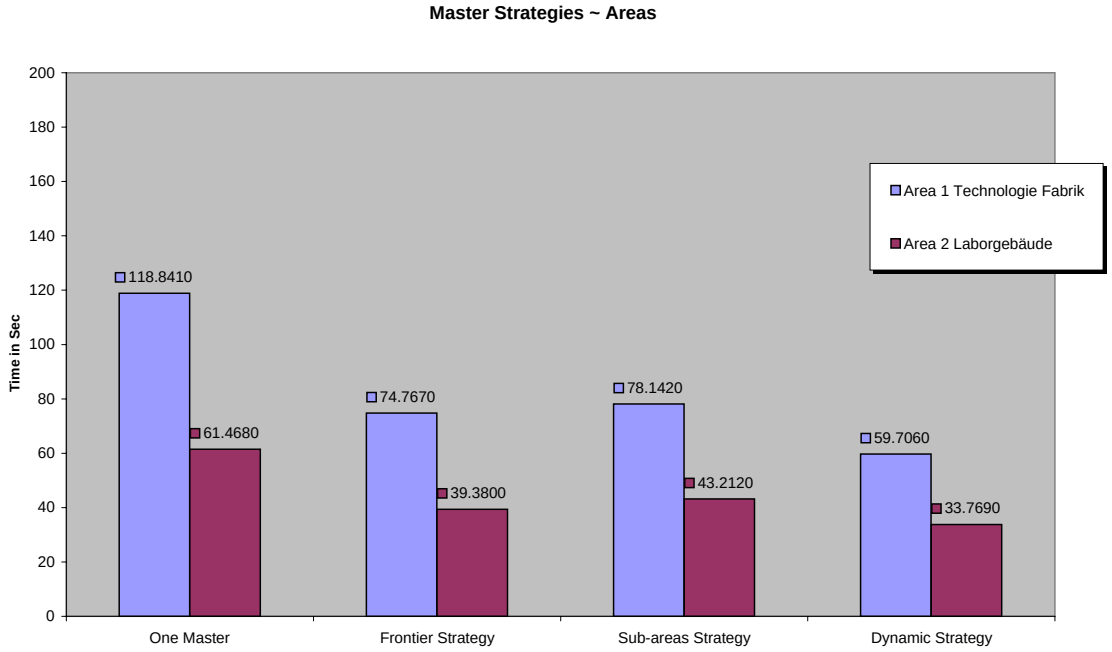


Figure 7.1: The performance of master cooperative strategies using two masters of the same speed

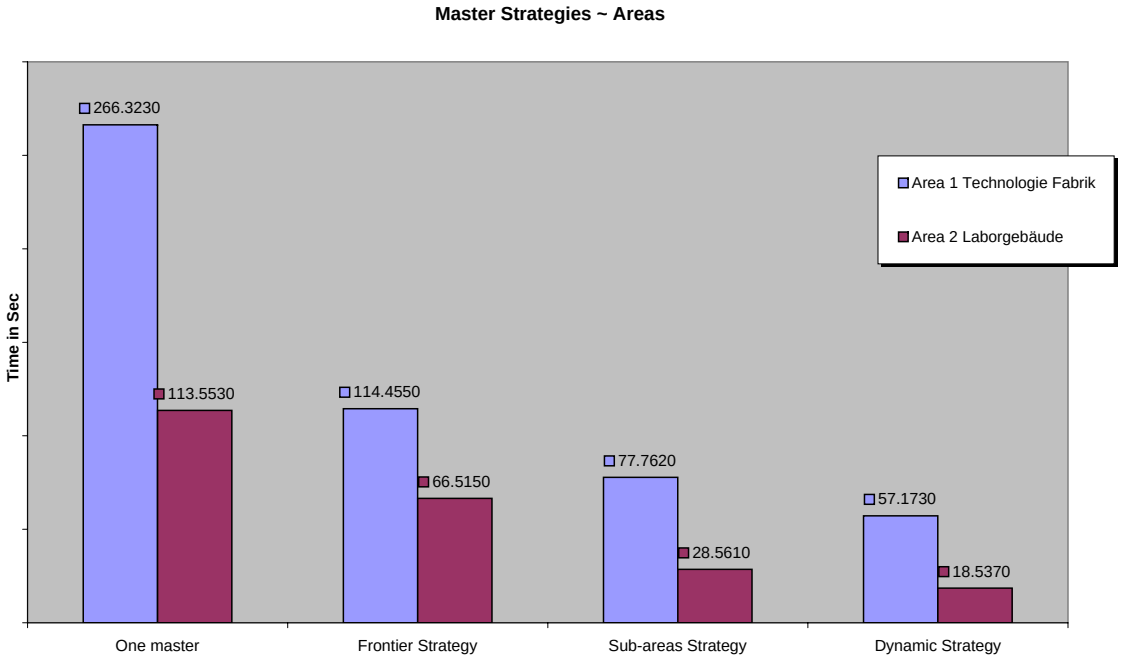


Figure 7.2: The performance of master cooperative strategies using two masters of different speed

out and the slave has to perform by itself the planning, navigation and scanning process. On the other hand, the second strategy pursues a distributed form in which the slaves themselves perform the coordination task as well.

To evaluate the coordination, one more version of each strategy was implemented as well.

They called centralized strategy version 1 and distributed strategy version 1. They were implemented with a weak coordination in order to evaluate our coordination mechanism.

The evaluation criteria for comparing the two strategies are their degree of autonomy, the time of exploration, the number of slaves, the area of the environment and the coordination mechanism.

To evaluate the degree of autonomy, we implemented the distributed strategy where the slaves themselves performed the planning, navigation, scanning process and the coordination process. These tasks give the slave a higher degree of autonomy, and let this strategy be more powerful, which is required in some applications to have a slave which can perform its task individually.

To evaluate the exploration time to scan the whole area, we used three slaves. Figure 7.3 presents the performance of each strategy. It can be noticed that the distributed strategy has a good performance compared to the centralized one. However, the time of performing the exploration process in the distributed strategy is a little bit higher, but this can be neglected because of its degree of autonomy.

As shown in figure 7.3, the evaluation test was performed on two different environment areas. It was found that, as the area of the environment became larger, more time was required in order to do the exploration. This is one of the main reasons to use a multiple robot system.

One more evaluation criteria on which affects the system performance is how many slaves should be used to achieve the task most efficiently. Different experiments were performed using up to 20 slaves in two different areas. It could be shown that, a higher number of slaves leads to a better system performance. However, the most efficient number of slaves depends on the environment which is to be explored. Figure 7.4 shows the system performance when using up to 20 slaves achieved in the Technologie Fabric environment. Within this environment, it can be noticed that more than 9 slaves were useless because the system performance does not improve any more.

To evaluate the coordination mechanism, we implemented the two versions of these strategies with a weak coordination. We found that with the weakest coordination the system performance becomes the worst. This can clearly be seen from figure 7.4. The time of the exploration and the stability of the system were affected by the weak coordination, such that the navigation time of the slaves was increased in order to allocate any further tasks.

As a result of investigating these criteria, we found that the distributed strategy achieves a good performance. It can be noticed from figures 7.3 and 7.4 that the distributed strategy has approximately the same performance of the centralized one, which is an indication of the success of using such a coordination mechanism.

7.2 Evaluation of the Feature Extraction Algorithm

The algorithm for extracting natural features was evaluated as well. Its performance was compared to the mean shift algorithm presented by Comaniciu and Meer [Comaniciu 02], and to the Indoor Space Scene Analysis (ISSA) by Rous [Rous 05], which is based on Hough transform techniques.

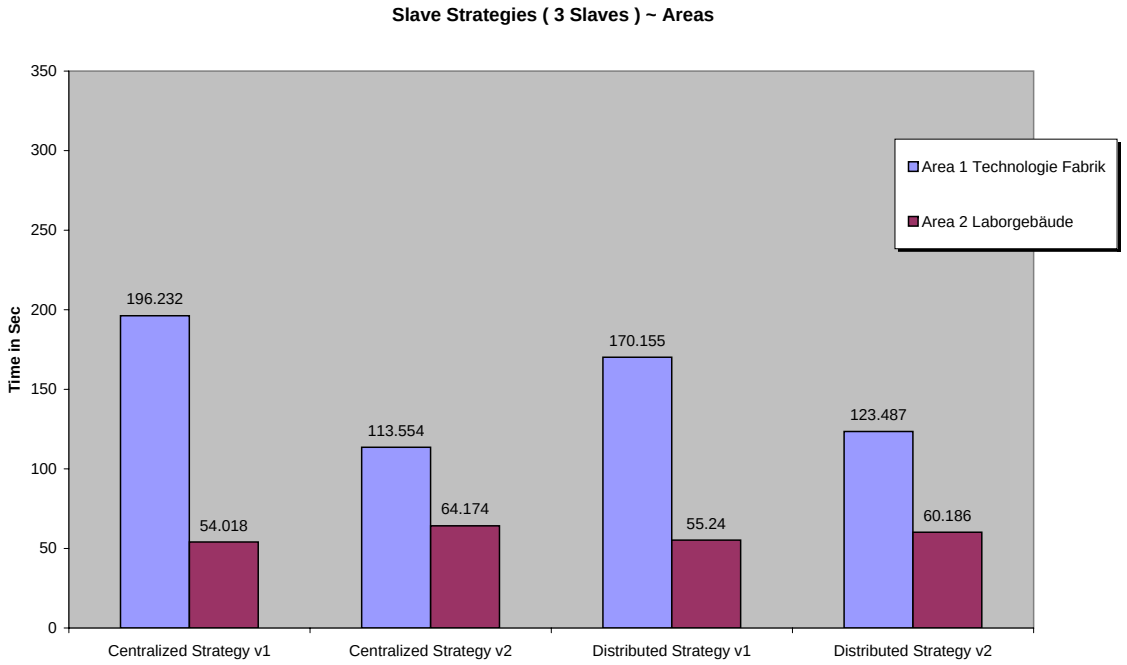


Figure 7.3: The performance of slave cooperative strategies using three slaves

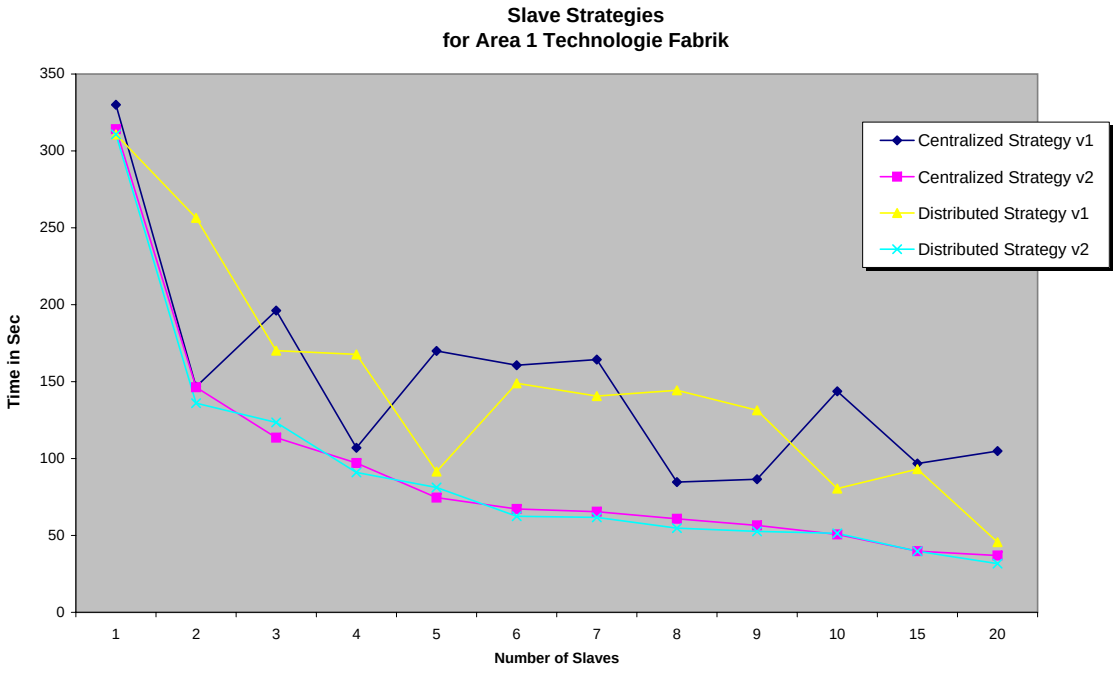


Figure 7.4: The performance of slave cooperative strategies using different numbers of slaves

Within this section, we first discuss the evaluation criteria in subsection 7.2.1. In subsection 7.2.2, we present the comparison and evaluation of these three algorithms.

7.2.1 Performance Criteria

Three evaluation criteria for feature extraction can be distinguished:

- Memory use and computational cost, which are determined by measuring the image segmentation time and the time need for object classification. Image size, frame rate of capture and computing time of the employed system are some important parameters which should be taken into account.
- The autonomy of the algorithm, which means the ability of the algorithm to perform the feature extraction without using a priori knowledge about the feature to be extracted.
- The quality and stability of the segmentation results.

To evaluate these criteria, all algorithms need to be implemented using the same modular controller architecture, and to be executed using the same computing system. Within our application, a mobile robot was used in order to perform the exploration of an unknown environment. In this task, a natural landmark is to be extracted in order to localize the mobile robot during the exploration process. The extraction process should satisfy the following requirements:

- The algorithm should be able to run online, i.e. the processing time of the algorithm should be within a few milliseconds.
- The algorithm should be able to extract the features from different places and under different conditions, which satisfies the stability requirements.

7.2.2 Evaluation and Comparison

The algorithms were implemented and tested using an MCA architecture in order to extract and classify three doors in an environment. Firstly, we evaluated the memory use and computation cost. Table 7.1 shows the computational cost of the algorithms for classifying the found features as doors. All algorithms were applied on an onboard computer of a mobile robot with Pentium III and 1.0GHz speed, with images of the size 640×480. The time required to execute the four steps of the feature extraction algorithm based on the mean shift algorithm is approximately 2.0 seconds. This algorithm also suffers from high memory requirements which reduces its efficiency. The execution time of the feature extraction algorithm based on ISSA using the Hough transform technique is with 0.56 seconds significantly faster than that of the mean shift procedure. On the other hand, our algorithm for feature extraction based on a prior knowledge, region growing and edge detection techniques was tested, and its execution time turned out to be the best in comparison with the other approaches.

Secondly, we tested the autonomy of each algorithm, and found that the mean shift procedure does not require any prior knowledge of the number of the clusters and does not constrain

No.	Method	Time in Second
1	Feature extraction algorithm based on mean shift procedure	2.0
2	Feature extraction algorithm based on Hough transform (ISSA)	0.56
3	Feature extraction algorithm based on region growing procedure	0.50

Table 7.1: Execution time of a feature extraction algorithm using different segmentation methods

the shape of the clusters. In the contrary to mean shift algorithm, a prior knowledge about the feature is required by the other algorithms in order to get a robust feature extraction.

The third criteria to be tested is the quality of the segmentation methods. We have found that the mean shift produces the highest quality of segmentation amongst the three algorithms as shown in section 6.3. As a result, the feature extraction algorithm based on the mean shift procedure performs the segmentation process autonomously without any prior knowledge of the feature. Its robustness and quality are good compared with the others. However, it has two disadvantages. Firstly, its computing time is too high, approximately four times more than the time needed by others. Secondly, its memory requirements are too high as well, which sometimes causes an unstable execution of the algorithm. For these reasons, the algorithm based on mean shift was not selected for the final application on the robot, since it is not executable online. That means, it is in effect not applicable for use in the field of robotic exploration.

Furthermore, we tested the stability of the other two algorithms. We found that the feature extraction algorithm based on Hough transform developed by Rous is not stable. That means, we need to adjust the threshold setting needed by both the canny edge detector and the Hough transform procedure when we capture a new scene. It performs a good system speed, however the correctness of the results have applied only on one scene. Therefore, it was also not selected as a basis of our feature extraction algorithm.

Finally, several experiments have been carried out along three different paths in order to test the stability of the feature extraction algorithm based on region growing. In the first experiment, the robot moved forward through the corridor. In this experiment the robot stopped every 5 meters to capture new images and extract the features. 76% of the feature were correctly detected and classified. 6% of the failures occurred due to an occlusion of the feature, whereas 18% of the failures occurred because the features were out of view at the start point of the experiment and could not be detected and classified. In the second experiment, the robot was directed to certain positions (selected for a good view point). New images were captured and new features were extracted. In this experiment, the algorithm correctly detected and classified 94% of the features. In the third experiment, the robot explored the environment following the exploration strategy algorithm. At each frontier region, new images were captured, in which new features were correctly extracted and classified. This experiment shows that the algorithm correctly detected and classified 85% of the features, only 15% of failures occurred due to an occlusion of the features at the frontier regions. Table 7.2 summarizes these performances.

	Correct Extraction	Occlusion Failure	Out of View Failure	Remarks
Experiment A	76%	6%	18%	5 times of capturing
Experiment B	94%	6%	–	3 times of capturing
Experiment C	85%	15%	–	3 times of capturing

Table 7.2: The stability of the feature extraction algorithm based on region growing in different experiments

7.3 Validity of the Algorithms

In order to evaluate the algorithms developed within the course of this work, some experiments were conducted in an environment of the size $60 \times 20m$. Two mobile robots were used with different configuration. The first mobile robot is equipped with stereo camera and laser scanner sensors as presented in chapter 3. This mobile robot was used as a master in order to get quickly the landmarks map, and to find places where should the slave work. Whereas the second mobile robot is equipped with a rotated scanner system (RoSi), as presented in chapter 3 and 6. This mobile robot was used as a slave in order to get a 3D scanning of a certain places in the environment, and to perform the fusion process to the data obtained by it in each step.

The aim of these experiments is to get a 3D digitalized model of a large indoor environment. Some original views of different scenes of this environment are shown in figure 7.8. The experiments were carried out in the first floor of the computer center building. This type of environment has a rectangular form, which can allow the robots to start from and end at the same place in order to close the mapping loop and to satisfy the algorithms used.

Two different experiments were conducted in this environment of the size $60 \times 20m$, with a robot speed of $0.2m/s$. Table 7.3 shows the dimension of the applied grid-based map, speed of the robots, and the time of the exploration done by the master and the slave.

Parameters	Values
Workspace	
Length × Width	$60m \times 20m$
Cell Size	$0.20m$
Robot Speed	$0.2m/s$
Time of the exploration done by the master	$13.845min$
Time of achieving all the tasks by the slave	$16.0min$

Table 7.3: Parameters of the robots, the grid-based map and the time of the exploration done in the experiments at the computer center building

The first experiment was carried out by the master. The master quickly builds a map of landmarks, finding some features to be used in the localization process by the slave when it is doing its tasks. The master makes a list of different positions in the environment as target places to be scanned by the slave, to get an accurate 3D map.

The robot was only directed through a corridor of this environment. Figure 7.5 shows different positions of the master during the exploration process. The robot completed its task in twelve steps within a time of 13.845 minutes. The feature-based and occupancy-based maps were used to represent the environment. Based on the grid-based representation, the motion planning was performed. The MCAGUI tools of the MCA framework were used to initialize the algorithm and to demonstrate the results of the mapping.

The task of the exploration was achieved in twelve steps. At each step, the robot applies the sequences demonstrated in figure 4.6. It starts with sensor sweeping to find the feature available around it and to detect where to go next, and it ends with navigating to the best target it found.

Figure 7.6 shows the results of all steps done be the master. At the first step for example, as the result of the exploration strategy, two frontier regions were detected at this step, which are represented by green colour. Frontier centroids were found for each region. The prospect were measured for each region. As a result, the master chooses the best the best prospect to continue achieving the task, and plans a path to this target, and then navigate to it as a second place of the exploration. This sequences will be repeated at each new place until no more regions to be explored. The last map in figure 7.6 shows the global map of the chosen environment, showing as well that there are no more regions to be explored.

The second experiment was carried out by the slave. The slave receives a target list to be scanned from the master. The list consists of twelve places in the environment, to get a 3D map of the chosen environment. As a result of the exploration algorithm, the slave applied the sequences demonstrated in figure 4.8 at each new place. Figure 7.7 shows different positions of the slave during the exploration process. The slave completed its tasks in twelve steps within a time of 16.0 minutes. Different forms of 3D data was built such as map including the texture information of the surrounding, or similar maps but without the environment's ceiling.

Within this experiment, the slave performed twelve laser scans. Parts of the resulting can be seen in figures 7.9, 7.10 and 7.11. From top to bottom, figure 7.10(a) presents the range image data obtained at the first location, figure 7.10(b) presents the fusion of the first scan with the result at the second and third position of scanning, and so on. In general, figures 7.9 and 7.10 show the same results of range image merging process at different scanning positions in the corridor at the computer center building, however figure 7.10 shows the results with cutting the environment's ceiling.

Figure 7.11 demonstrates the final results of range image merging process at different scanning positions in the corridor at the computer center building in different cases, such as in the case of point clouds, in the case of cutting the ceiling of corridor or in the case of a triangulated point clouds.

It can be concluded from these experiments that a building of a 3D map is successful, starting by the work done by the master, which builds an accurate landmarks map that was used in

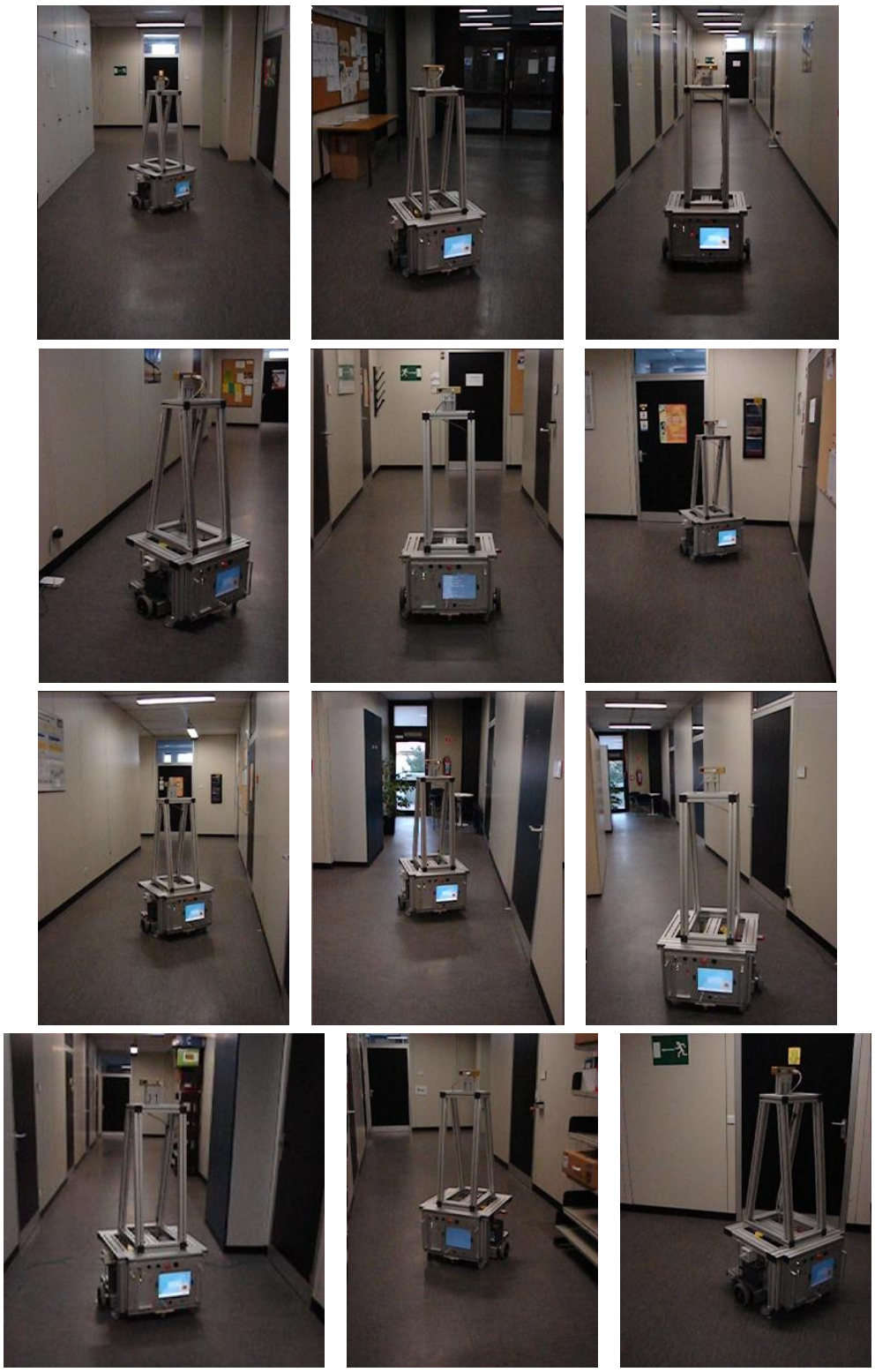


Figure 7.5: Different positions of the master during the exploration process.

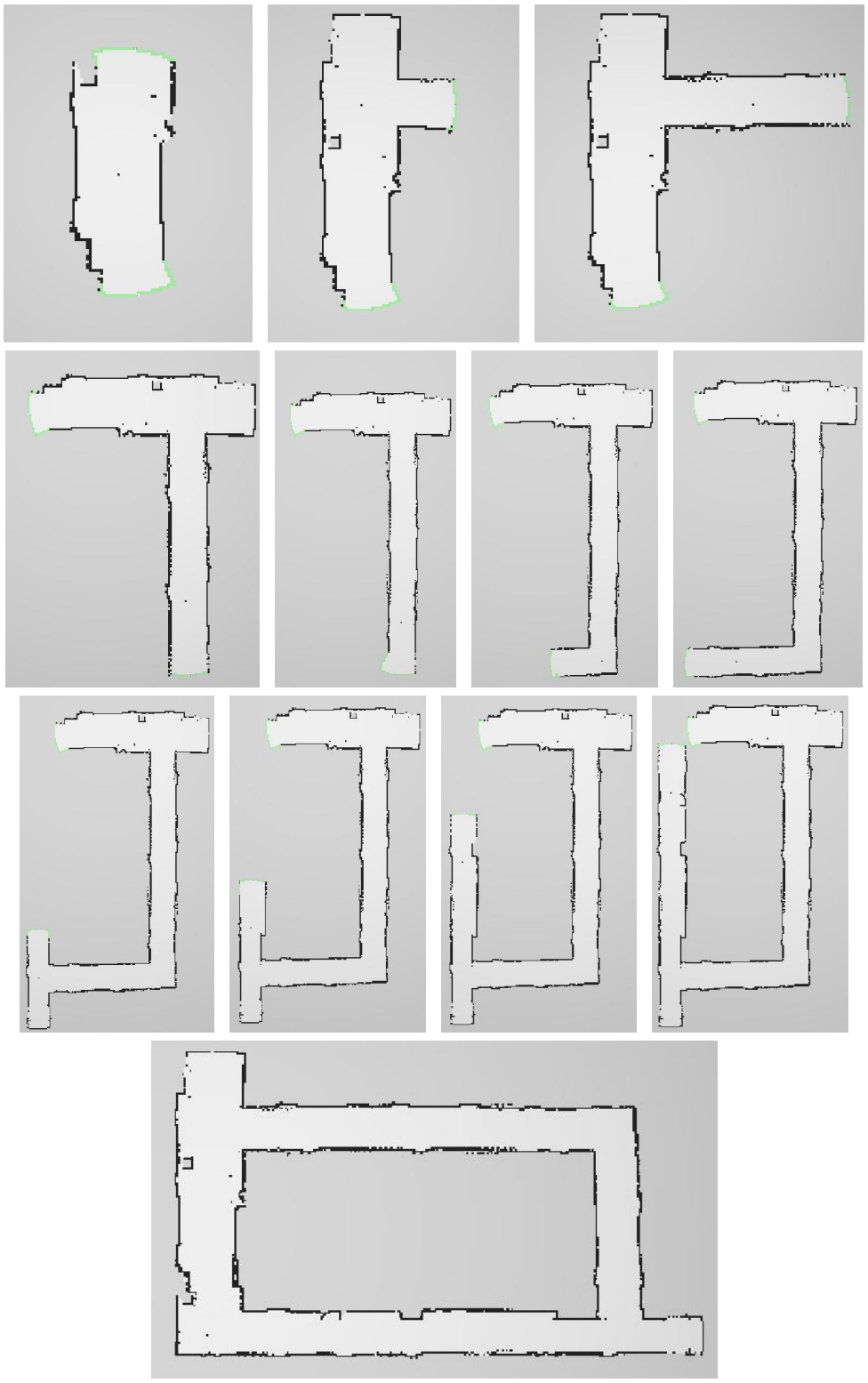


Figure 7.6: The results of the exploration algorithm done by the master within 12 steps of motion control.

the localization process by the slave. Ended by the work done by the slave, which gets an accurate 3D map of certain places in the chosen environment, and merges them into a model of this environment into three different cases. It is clear from these maps that the map loop is closed in an efficient way.

7.4 Investigation of the Environmental Representation

In order to evaluate quality of the maps obtained within the course of this work, some experiments were conducted under the same configuration presented in chapter 4. Within these experiments, both robots try to capture the same feature knowing its location from different positions. The aim of these experiments is to get how the obtained maps are accurate depending only on the sensor type and its resolution.

Within the work, three environmental representations were used. Firstly, the grid-based map was obtained by using 2D laser scanner. The accuracy of this map was around 10 cm, which is the cell size used within the algorithm. As presented in chapter 4, this type of map is only needed to known where the robot can go next step and how it can do by performing the planning. Therefore, we do not need a good accuracy within this type of the map.

A natural landmark was extracted using a stereo camera in the case of the feature-based map, whereas the RoSi system was used to build a 3D environmental model. Table 7.4 shows the result of one experiment in order to get a location of a landmark. It can be conducted from this experiment how the maps are accurate. Table 7.5 presents the accuracy of the maps in three different cases.

	Original	Captured by stereo camera	Captured by RoSi system
Feature location	$4.880m$	$4.887m$	$4.846m$
Difference	0.0	$\approx 0.007m$	$\approx 0.034m$

Table 7.4: One example of detecting a position of a known feature

	Grid-based map	Feature-based map	3D environmental model
Sesnor type	2D laser scanner	Stereo camera	RoSi system
Map accuracy	$10cm$ (Cell size)	$\approx 7mm$	$\approx 3.4cm$

Table 7.5: The environmental maps and their accuracy

7.5 Conclusion

This chapter presented the performance of a master-slave system developed within the course of this work considering two main criteria: the time of exploration and the coordination efficiency of the system.

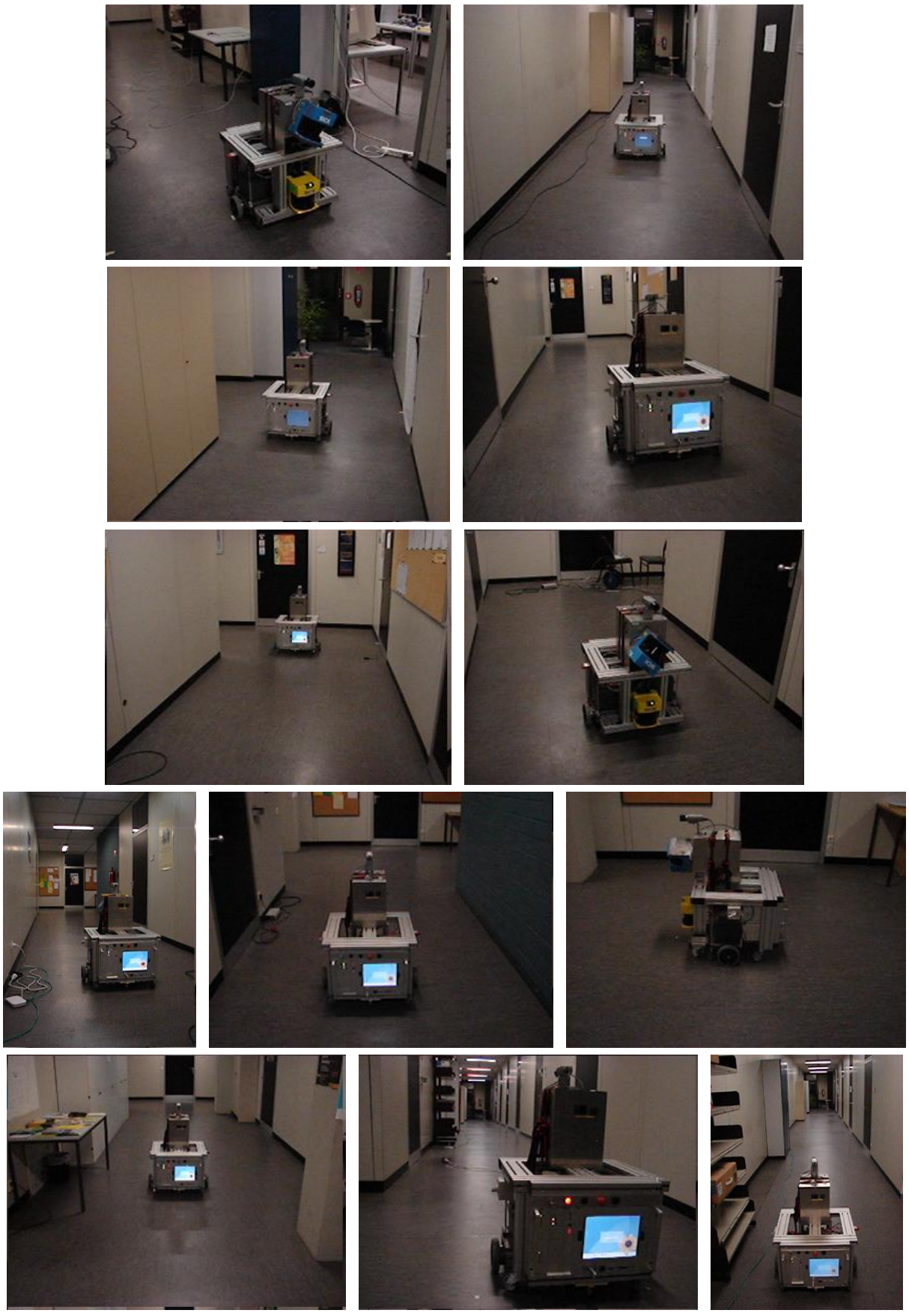


Figure 7.7: The slave is at different places in the environment chosen by the master in order to execute the task of a 3D scanning.

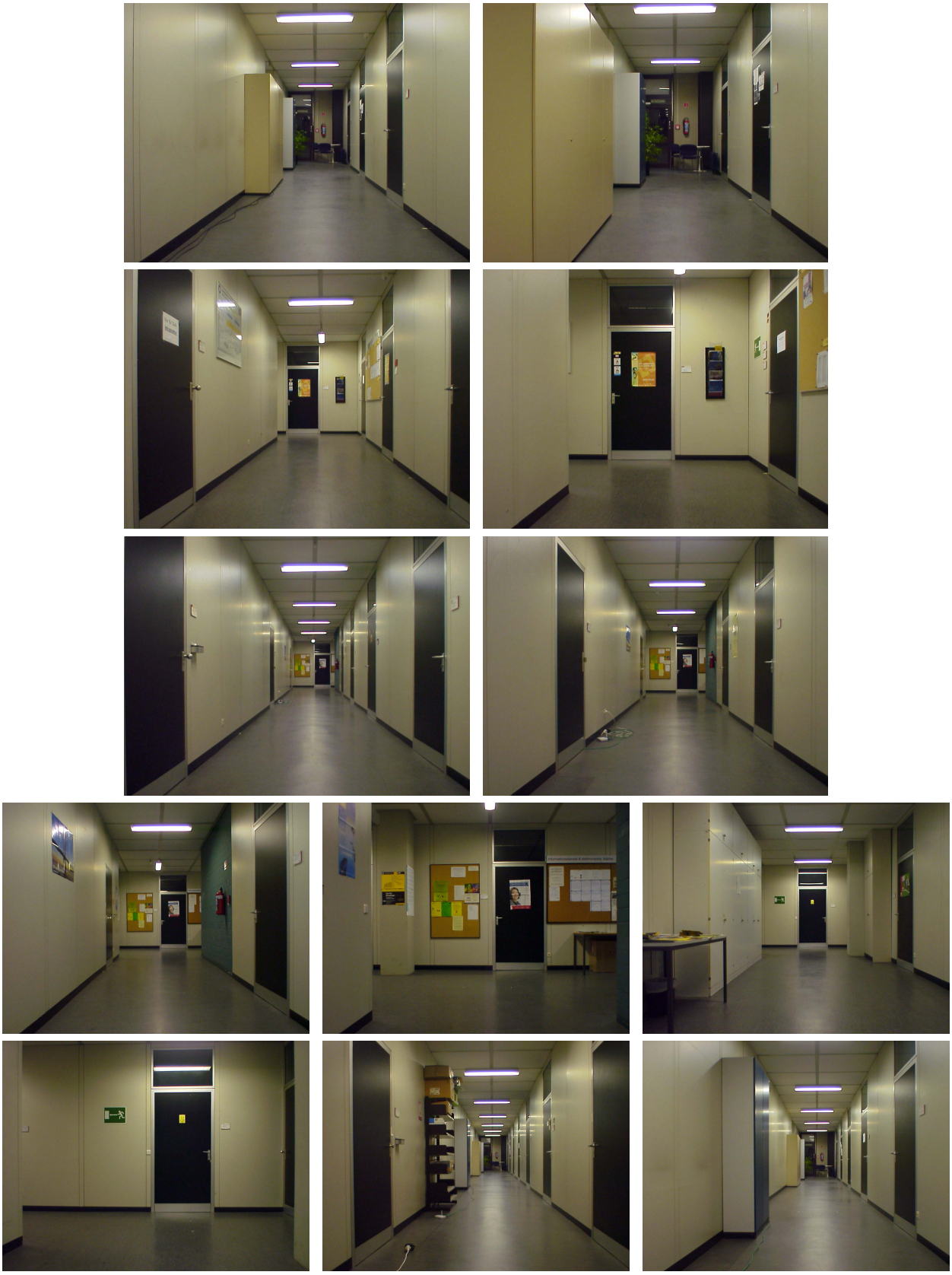
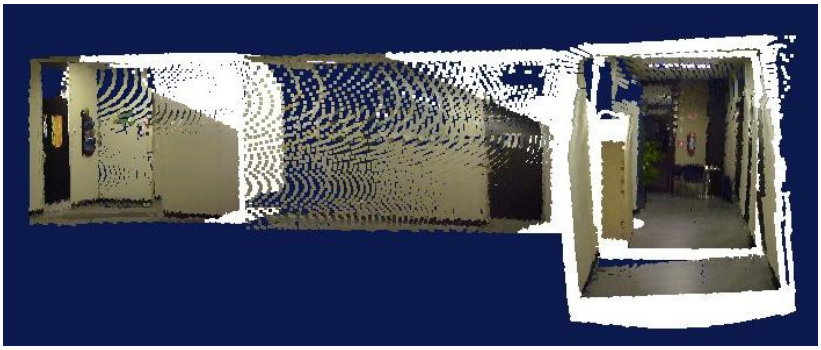
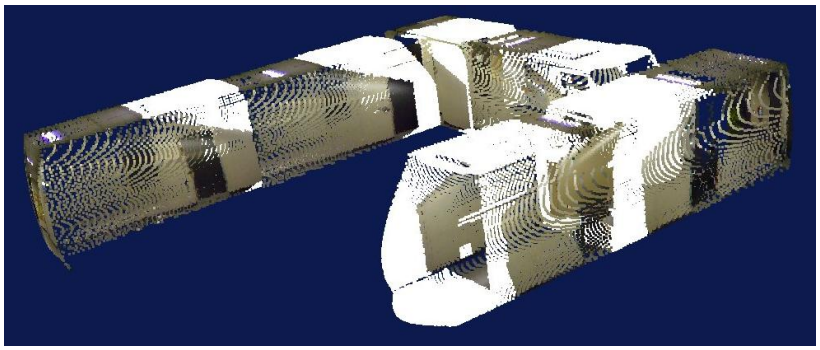


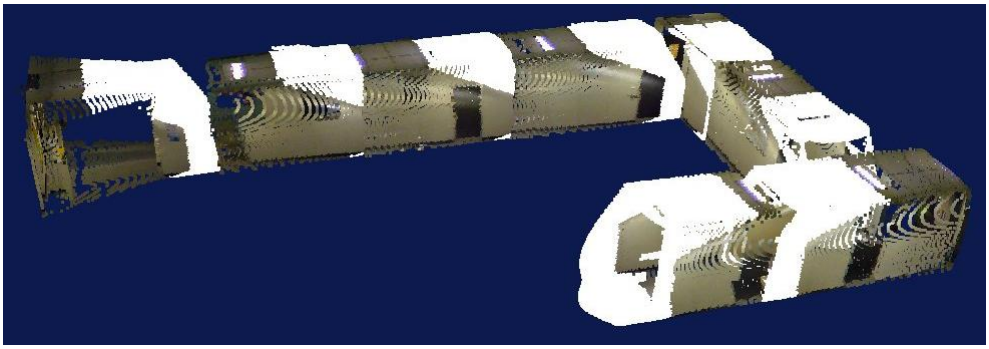
Figure 7.8: Original views of different scenes to be scanned by the RoSi system, these scenes are captured by the digital camera equipped with RoSi system at different position of the slave.



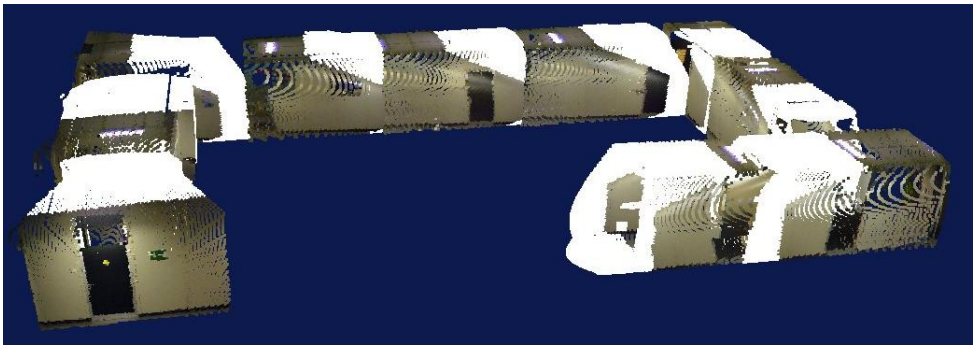
(a)



(b)



(c)



(d)

Figure 7.9: Results of range image merging at different scanning positions in the corridor at the computer center building: (a) merging 4 range images, (b) merging 6 range images, (c) merging 8 range images, and (d) merging 10 range images.

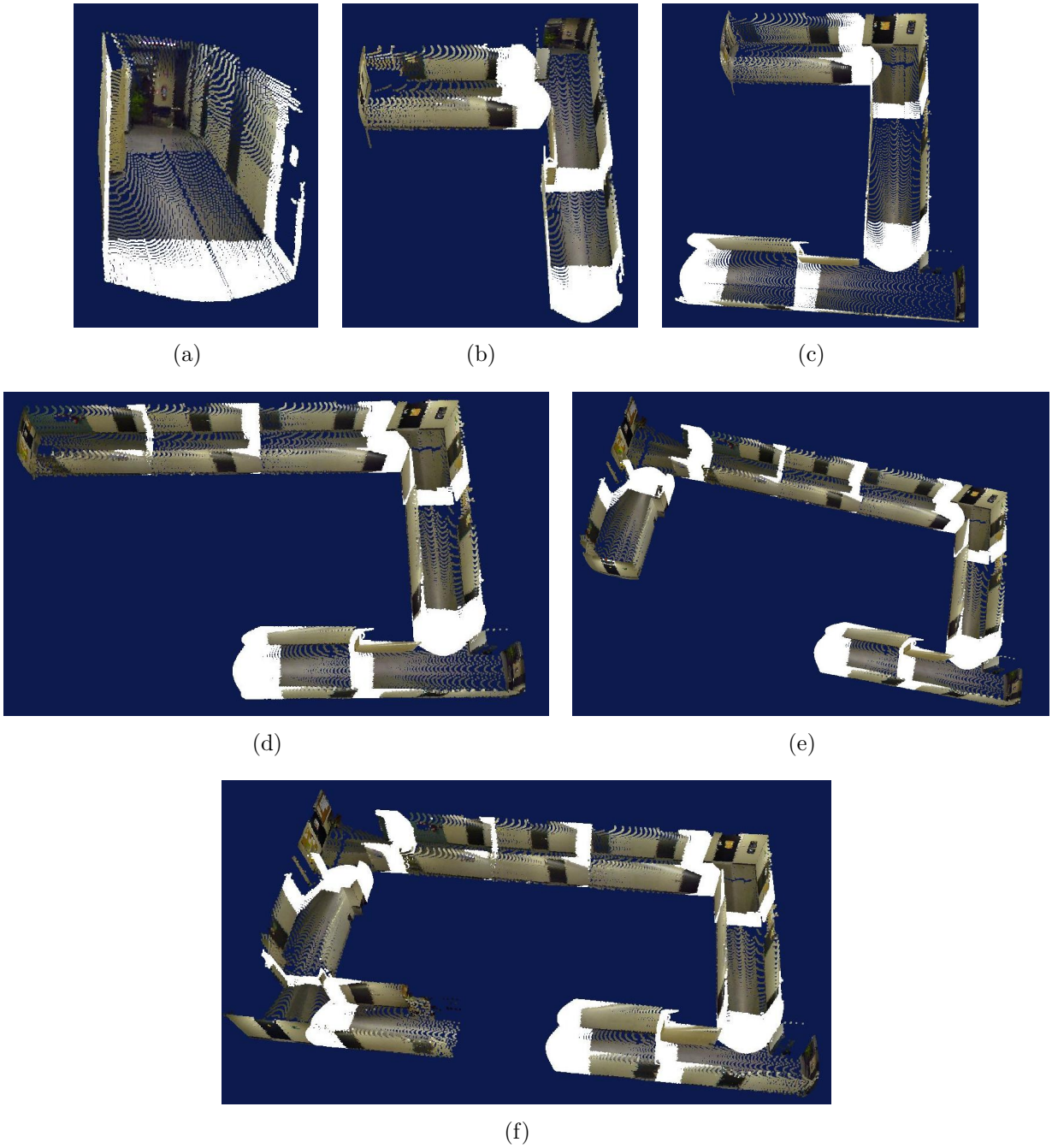
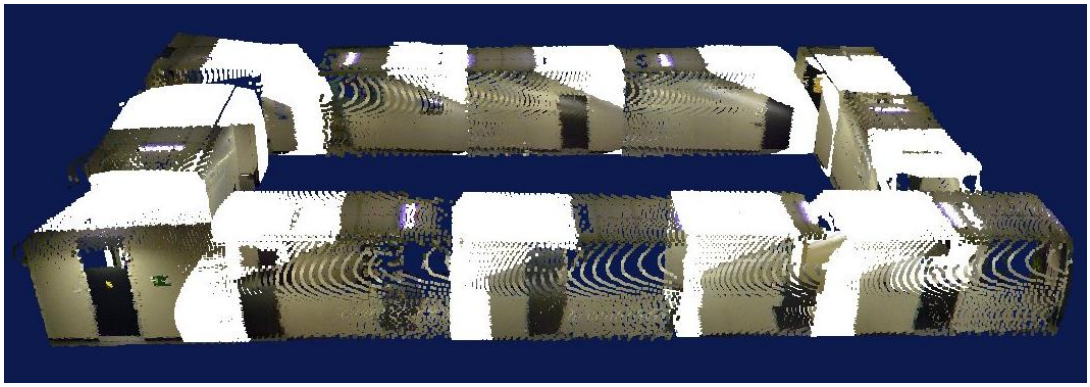
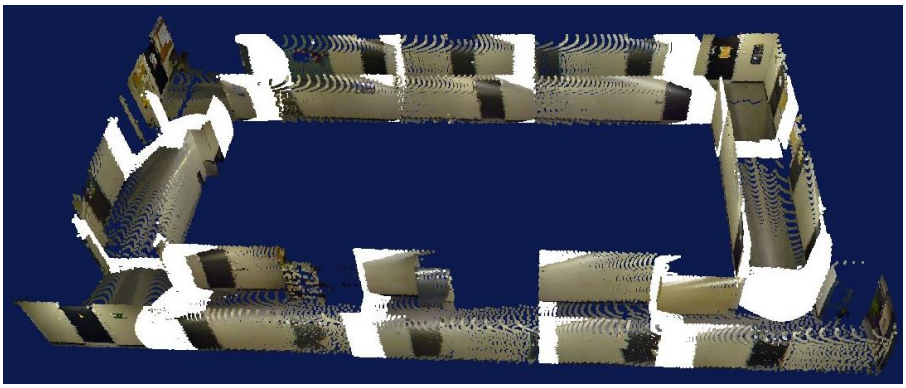


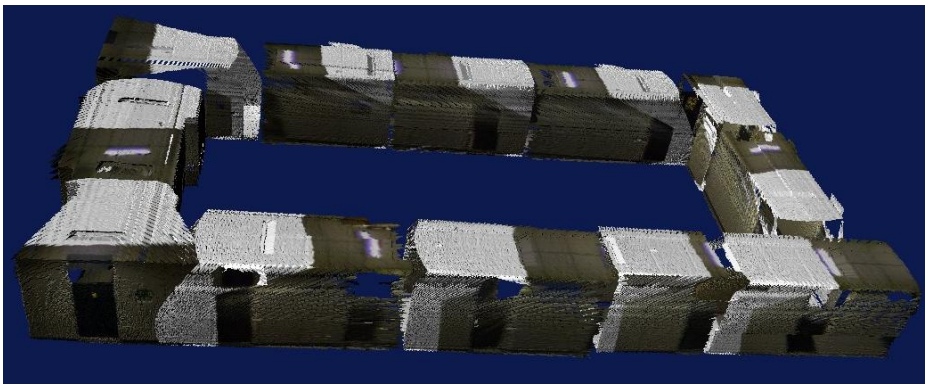
Figure 7.10: Results of range image merging at different scanning positions in the corridor at the computer center building showing the scene without its ceiling: (a) the first scanning result by the slave, (b) merging 3 range images, (c) merging 5 range images, (d) merging 7 range images, (e) merging 9 range images, and (d) merging 11 range images.



(a)



(b)



(c)

Figure 7.11: The final results of range image merging at different scanning positions in the corridor at the computer center building in different cases: (a) in the case of point clouds, (b) in the case of cutting the ceiling of the corridor, and (c) in the case of a triangulated point clouds.

The first section introduced the performance evaluation of the master-slave system, concerning in particular with the cooperation between master and slaves. The performance of different strategies were evaluated taking into account different criteria such as the degree of cooperation, the coordination efficiency, the time of the exploration, and the size of the robotic team. The evaluation performance was separately discussed for each case (master and slave case).

The second section introduced the performance evaluation of the feature extraction algorithm used within the course of this work. Three different types of segmentation methods were evaluated in order to test our algorithm. The stability of this algorithm was tested as well by different experiments in different situations in an indoor environment.

The third section concerned with the validity of the algorithms developed within the course of this work by doing some experiments in a complex environment. Two different experiments were carried out in the first floor of the computer center building. As a general conclusion, the 3D map is obtained in a way that the map is closed in an efficient way.

Chapter 8

Conclusions and Further Work

This final chapter summarizes the research presented in the previous chapters, concentrating on the contribution of the current work as stated in section 1.3. After a brief summary, it discusses the presented contributions with regard to their advantages and limitations, and points out perspectives on future research.

8.1 Summary

The main goal of this thesis was to autonomously build a 3D map of indoor environments within a good exploration time by using multi-mobile robots. During the process, the exploration algorithm chooses appropriate motion commands to let the robots achieve the tasks as quickly as possible. A good collaboration among a team of robots was achieved in order to let them efficiently solve the exploration task. As a result, the overall time to complete the exploration mission was reduced. The coordination among the robots was investigated in particular so that they efficiently distribute themselves over the environment avoiding redundant work and reducing the risk of interference between the vehicles.

Within the course of this work, a solution to the exploration task was implemented and tested on multiple real robots. Furthermore, a multi-robot simulator was also implemented to evaluate and test the system performance. To conclude this work, we highlight the main achievements of this thesis in form of the work contribution to the field of autonomous exploration in terms of the problem statement constituted in section 1.2.

The first achievement of this thesis is the design of a framework for 3D mapping using multi-mobile robots. In this context, coordination is an essential characteristics of any task-achieving multi-robot system. Thus, the framework architecture was designed to allow the development of multi-robot systems that can manage the coordination problem efficiently.

Therefore, an approach to distribute the robots over the environment was implemented in order to avoid redundant work. A master-slave approach was used to coordinate them, with a hierarchy of one master and n slaves to finish exploration tasks efficiently. By using a simulation package, this hierarchy was extended to two masters with n slaves assigned to each of them, which reduced the time of exploration.

An important benefit from this architecture was the ability to handle different sensor capabilities of the master and the slave. This heterogeneity enhances the team's robustness. For example, the master has a stereo camera which can capture up to 30 frames per second, whereas the slave has a RoSi system, which requires more than 7 seconds to perform a single scan. Because of that, the master can quickly built up a feature-based map, and direct its team to appropriate places to get a 3D map. Another benefit from this structure was the ability to handle different types of data because of the different sensor types and representations used by the robots.

The second achievement of this thesis is the development of a motion command algorithm, which lets the robots move within the environment by choosing a new exploration viewpoint based on their current map. The strategy implemented, makes the robots move to a new unexplored place based on certain prospect values of this place.

This exploration strategy was elaborated both for single and for multiple robots. For a single robot, two prospect values are used: The first prospect is the natural feature quality available at the next target, whereas the second prospect value is the navigation cost to this target. For multiple robots, besides the two prospects mentioned before, a coordination prospect function was also introduced in order to distribute the robots efficiently over the environment. Experiments with a single mobile robot equipped with a stereo-vision system successfully validated the proposed exploration method. It was also demonstrated experimentally with multiple robots that the exploration time improves with our exploration strategy.

The third achievement of this thesis is the development of cooperative strategies among the robots in order to explore environments using a master-slave approach. A detailed simulative analysis of different cooperative strategies for both master and slave was performed, in particular with regard to identifying the advantages of using multi-mobile robots and a master-slave hierarchy. This multi-robot simulator was designed taking into account a model of the robots and their sensors. Two types of environment were also modelled to perform this simulation.

As presented in the problem statement, the SLAM algorithm suffers from the computational and storage requirements in order to run the algorithm online. The sub-areas strategy was illustrated, where partitioning the whole area into sub-areas improved the computational and storage requirements. In this strategy, each robot selects a limited area to be explored. When it finishes exploring this area, it starts exploring a new one. This strategy improves the degree of cooperation among the masters and leads to considerable results compared with the frontier strategy.

The distributed strategy was implemented to be used by the slaves. It could be shown that, within this strategy, the slaves performed the scanning tasks very efficiently compared to the centralized strategy. This strategy satisfies the requirements of applications, in which a higher degree of autonomy is required. In this strategy, the slaves perform the coordination, planning, navigation and scanning by themselves. Only a few pieces of information are required from their master in order to perform the task, which improves the communication load.

The concepts and prospect values of the motion control algorithm developed in chapter 4 were also tested within this simulation. In these experiments, the coordination was handled

in an efficient manner without any external assistance. This ensures that in the case of a single point of failure the performance of the team only degrades gracefully, and each robot still continues the task. Thus, the solution is fault tolerant.

To apply these cooperative strategies in the real world, some communication protocols were implemented in order to interchange the measurement data among the robots. Furthermore, a way of passive communication was presented in order to let a robot estimate the position and estimation of its team using different sensor capabilities.

The fourth achievement of this thesis is the evaluation of the system performance, in which meaningful experiments were carried out for most aspects of the work in a real environment by using two mobile robots equipped with different sensor capabilities onboard. As the main goal is the time of the exploration, different methods were proposed to evaluate the performance of the system based on the time of achieving the tasks. On the other hand, some simulative experiments were also conducted to evaluate the benefit from different cooperative strategies. Several evaluations criteria were applied. Additionally, two different areas and different number of slaves were used to perform the simulations. As a result, the optimal size of the team depends on the size of the area and the number of tasks to be done, and after reaching some limit, even more slaves will be useless.

Finally, some other algorithms were implemented to achieve the goal stated above. A feature extraction algorithm was developed in order to extract natural landmarks in the environment. Two sensor capabilities were used to perform this algorithm. Experiments were conducted to validate this algorithm, and it is found to be stable over different places in our indoor environment. This algorithm performs the following steps: data acquisition, pre-processing, segmentation and feature classification. Such an approach allows to describe the measurement process for different part of the environment and to avoid the data association of the extracted features of different places.

A fusion of different range image data was performed, where the point clouds of different places captured by different slaves were received by the master and fused into a global map of the environment.

A reliable algorithm for simultaneous localization and map building in an unknown environment was also achieved by using information from the measurements of arbitrary features and the robot odometry. Within this algorithm, the extended Kalman filter was used to match the features in order to recover an accurate position estimate of the robot.

8.2 Discussion and Suggestions for Further Work

The previous section summarized the research pursued within this thesis, and presented their achievements. Besides the advantages, the contributions of this thesis also have some limitations. while discussing them, possible extensions and some suggestions for further work are presented in this section.

The most obvious limitation is concerned with the communication. We have assumed within our environment a stable wireless network, which can be accessed in any place of the environment. However, in an unknown environment, this assumption may not hold, and the

communication might be restricted to only a few meters. In this case, the robots would have to remain together in order to avoid covering the same area multiple times. A simple approach to solve this problem was introduced by Rekleitis [Rekleitis 04], in which the robots operate under the restriction that communication between two robots is available only when they are within the line of sight of each other. This approach can be applied to our framework using the sub-areas strategy, such that two masters follow some procedure that allows them to always explore two adjacent sub-areas and to wait for their slaves to finish their tasks within this area. As a result, the time of exploration will be increased.

Another factor which is not addressed within our work is communication with limited bandwidth. Meier and his colleges [Meier 05] presented a strategy in which the robots only transmit changes and refinements of the learned map among each other in order to achieve the explorations, since they used a communication with limited bandwidth. Within our work, we have used an edge form to exchange the grid-based map among the robots which improves the use of communication bandwidth. However, the range image data exchanged between the master and its slaves also needs to be optimized. Further work could concentrate on how to exchange the range image data in terms of polygons in order to save more communication bandwidth.

Although the registration process is an important issue, especially if an accurate 3D model of an indoor environment is required, it is out of the scope of this thesis, and thus it is not addressed within the course of this work. Further work on this topic should be carried out to improve the quality of the 3D model of the explored indoor environments.

In the cooperative strategies, the dynamic strategy was implemented in such a way that two master explore the environment starting from opposite corners of this environment. We found that the dynamic strategy has a better exploration time, the best cooperation degree and the best partitioning result of the whole area, because the robots are cooperating all the time. This strategy can be efficiently applied in order to perform a surveillance mission in which a partially known environment is available. In such a condition, the dynamic strategy can be guaranteed to improve the performance of the surveillance mission.

The reliability of the proposed system was demonstrated by extensive experimentation on a static indoor environment. In the future, our system could be generalized to also be applicable on dynamic and on outdoor environment. In this case, the feature extraction algorithm should be generalized in order to extract and classify natural landmarks of an outdoor scene to be used in the SLAM algorithm.

Appendix A

Basic Concepts of SLAM Theory

Early work in this area tended to focus on map making and localization as separate and unrelated problems. The problem of localization given a map of the environment or estimating the map knowing the vehicle position has been addressed and solved using a number of different approaches. A more difficult problem is when both, the map and the vehicle position are unknown. In this case the mobile robot starts at an unknown location in an unknown environment and proceeds to incrementally building up a navigation map of the environment while simultaneously using this map to update its location.

Variation of the simultaneous localization and map building (SLAM) problem in general have been addressed by different authors, e.g. [Dissanayake 01] [Nevado 04] and [Thrun 04]. A solution to the SLAM problem using stereo camera, was introduced by Davison [Davison 02]. He achieved good results by detecting and tracking suitable landmark features during goal-directed navigation. Within our work, we have adopted his algorithm in order to perform the SLAM process. The formulation of the mathematical framework for SLAM was derived to be suited to our sensors and to our feature extraction algorithm.

The algorithm adopts the Kalman filter-based approach, which directly provides both a recursive solution to the navigation problem and a means of computing consistent estimates for the uncertainty in vehicle and map landmark locations on the basis of the statistical models for vehicle motion and relative landmark observation. Dissanayake and his colleges [Dissanayake 01] have shown that it is possible for an autonomous vehicle to start at an unknown location in an unknown environment using a Kalman filter-based approach and collecting relative observations only, to incrementally build a perfect map of the world and to simultaneously compute a bounded estimate of the vehicle location.

Using vision-based SLAM, we propose an estimation process based on the Extended Kalman Filter (EKF) in order to accommodate a three-dimensional world coordinate model. This is done by estimating the range, the bearing and the elevation data of the landmarks with respect to the vehicle. A procedure should be provided in order to let the robot estimates its position based on the predication of some landmark locations in Cartesian coordination, and on its known kinematics model. Because of that, we present in this section a technique to achieve this. first, we present the probabilistic SLAM framework, then the system states and its covariance. Then, the vehicle and landmark models are presented. Furthermore, the estimation process is presented showing the prediction and update process of the filter.

A.1 Probabilistic Framework

The Bayesian probability theory was described by Bruyninckx [Bruyninkx 02]. Their work has made clear that the Bayesian probability theory is a very attractive framework and is one of the major theoretical and practical frameworks for reasoning and decision making under uncertainty. One of the major applications of the Bayesian probability theory is an autonomously navigating robot. Thrun et. al. [Fox 98] have used a Markov Localization framework which is one example of a Bayesian framework that estimates the robot's location combing information from multiple sensors. From this point of view one can say that a general localization problem or a general mapping problem can be described as a Bayesian estimation problem.

A.1.1 Definitions

To understand, how SLAM approaches are working, we first by introduce the notation and formalize the acting and sensing of a robot in probabilistic models. Consider a mobile robot moving through an environment taking relative observations of a number of unknown landmarks using a sensor located on the robot. At time instant k, the following quantities are defined:

- x_k : the state vector describing the pose of the vehicle
- u_k : the control vector applied at time $k-1$
- m_i : the location of the i_{th} landmark
- z_{ik} : observation taken from the vehicle of the location of the i_{th} landmark at time k

In addition, the following sets are also defined:

- $X_{0:k} = \{x_1, x_2, ..., x_k\} = \{X_{0:k-1}, x_k\}$: the history of vehicle locations
- $U_{0:k} = \{u_1, u_2, ..., u_k\} = \{U_{0:k-1}, u_k\}$: the history of control inputs
- $Z_{0:k} = \{z_1, z_2, ..., z_k\} = \{Z_{0:k-1}, z_k\}$: the set of all landmark observations

A.1.2 SLAM Probabilistic

The probabilistic form of the SLAM problem requires the computation of the probability distribution

$$P\left(x_k, m \mid Z_{0:k}, U_{0:k}, x_0\right) \tag{A.1}$$

for all times k. This probability distribution describes the joint posterior density of the landmark locations and the vehicle state (at time k) given the recorded observations and

control inputs up to and including time k together with the initial state of the vehicle. In general, a recursive solution to the SLAM problem is desirable. Starting with an estimate for the distribution $P(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1})$ at time $k-1$, the joint posterior, following a control u_k and observation z_k, is computed using Bayes theorem.

The SLAM algorithm is implemented in a standard two-step recursive form. In the time update step, a mobile robot receives its relative observation to get the belief where it is. Simultaneously the absolute observation is received to get the map estimate. By applying the total probability theory we can express the location of the robot and the map of the environment as well by a transition density as follows:

$$P(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0) = \int P(x_k \mid x_{k-1}, u_k)\, P(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0)\, dx_{k-1} \quad \text{(A.2)}$$

In the second step, the resulting absolute measurement is to be incorporated into the belief of the mobile robot to give it the most up-to-date map and location estimate. By Bayes' theorem, the observation is updated in the next step to get the posterior belief of the map and location estimate as follows:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k \mid x_k, m)\, P(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k \mid Z_{0:k-1}, U_{0:k})} \quad \text{(A.3)}$$

where $P(z_k \mid x_k, m)$ is the sensor model and $P(x_k \mid x_{k-1}, u_k)$ is the motion model in terms of the conditional probability. Equation A.2 and A.3 have a familiar form called "*recursive Bayesian estimation*", where many people have been motivated to find computationally tractable ways to perform these Bayesian information processing operations. One way to deal with the computational complexity of beliefs over continuous spaces is by representing the beliefs as a parameterized continuous function. The Extended Kalman filter is one of the most efficient Bayesian estimators, which is a means of calculating the beliefs that are represented by Gaussian, and it would appear to be an excellent way in which to implement these equations.

The map building problem may be formulated as computing the conditional probability density $P(m \mid X_{0:k}, Z_{0:k}, U_{0:k})$. This assumes that the location of the vehicle x_k is known at all times, subject to knowledge of initial location. A map m is then constructed by fusing observations from different locations. Conversely, the localization problem may be formulated as computing the probability distribution $P(x_k \mid Z_{0:k}, U_{0:k}, m)$. This assumes that the landmark locations are known with certainty, and the objective is to compute an estimate of vehicle location with respect to these landmarks.

A.2 System States

The setting for the SLAM problem is that of a vehicle with a known kinematics model, starting at an unknown location, moving through an environment containing a population of features or landmarks. The vehicle is equipped with a sensor that can take measurements of the relative location between an individual landmark and the vehicle itself as shown in figure A.1.

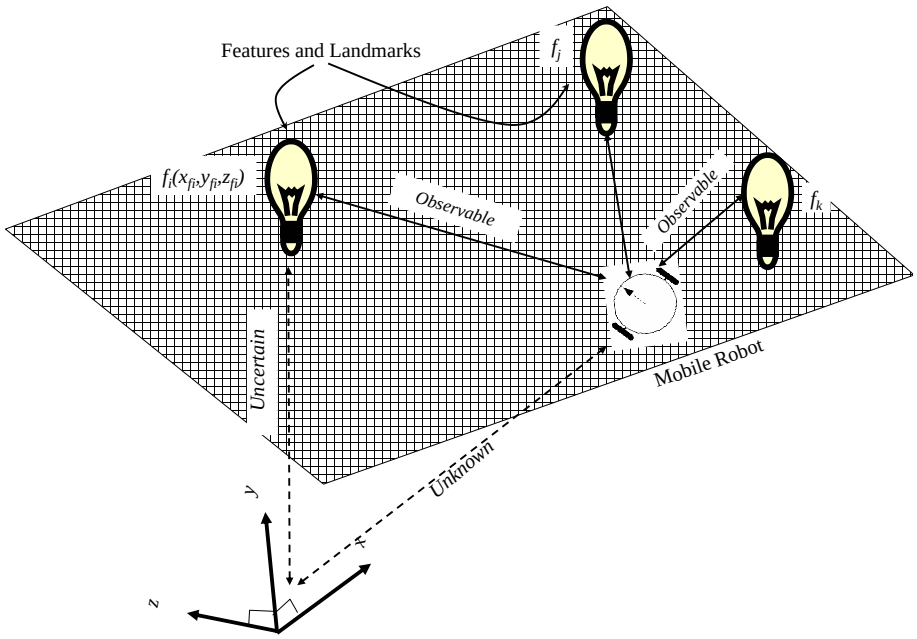


Figure A.1: The SLAM problem: A vehicle takes relative observations of environment landmarks. The absolute location of landmarks and vehicle are unknown.

A.2.1 State Vector

The vehicle state at time k can be uniquely determined by its position and orientation in space. The vehicle state is therefore defined by

$$X_v(k) = \begin{pmatrix} x_v(k) & y_v(k) & \theta_v(k) \end{pmatrix}^T$$

where x_v, y_v denote the location of the local robot coordinate system with respect to some global coordinate frame and θ_v is the heading with reference to the x-axis.

A simple state vector $f_i(k)$ of a feature i at time k is also defined as shown in figure A.1, consisting of a Cartesian position (x_{fi}, y_{fi}, z_{fi}) in the same reference frame as used for the vehicle,

$$f_i(k) = \begin{pmatrix} x_{fi} & y_{fi} & z_{fi} \end{pmatrix}^T$$

and the map vector is defined by

$$m(k) = \begin{pmatrix} f_i & f_j & f_k & \dots \end{pmatrix}^T$$

The augmented state vector is formed containing both the state of the vehicle and the state of all landmark locations, and it is denoted by,

$$X(k) = \begin{pmatrix} x_v^T(k) & f_i^T & f_j^T & f_k^T & \dots \end{pmatrix}^T$$

A.2.2 Covariance

The uncertainty in the estimates of the locations of the robot and the landmarks are stored in the covariance matrix P as follows,

$$P = \begin{bmatrix} P_{vv} & P_{vf_i} & P_{vf_j} & P_{vf_k} & \dots \\ P_{f_iv} & P_{f_if_i} & P_{f_if_j} & P_{f_if_k} & \dots \\ P_{f_jv} & P_{f_jf_i} & P_{f_jf_j} & P_{f_jf_k} & \dots \\ P_{f_kv} & P_{f_kf_i} & P_{f_kf_j} & P_{f_kf_k} & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

where P_{vv} is the vehicle covariance, P_{vf_i} is the covariance between the i^{th} landmark and the vehicle, and $P_{f_if_j}$ is the covariance between the i^{th} and j^{th} landmarks. The covariance matrix P is symmetric, with size $3\,(n+1) \times 3\,(n+1)$, where n is the number of known landmarks. This matrix P changes in dimension as landmarks are added or deleted from the map.

A.3 The Vehicle and Landmark Models

Figure A.2 illustrates the manner how the mobile robot is moving, in which the vehicle's location is represented on the ground. It is specified by the coordinates (x, y, θ), where x and y are the world frame coordinates of the rear centre, and θ is the robot's orientation relative to the world x axis. Figure A.3 shows a schematic observation kinematics diagram of the robot in the process of observing one of these landmarks. In this case, the vehicle is equipped with a stereo camera that provides us with a measurement which can be used to calculate the range z_r, the bearing z_θ and the elevation z_ϕ to an observed feature f_i relative to the vehicle. Both figures will be used to explain the vehicle , sensor and landmark models.

A.3.1 Vehicle Model

The mobile robot is moving by using a differential drive mode based on two main wheels, each of which is attached to its own motor. To construct a simple model of the constraints that arise from the differential drive, only the distance D between the two wheels is necessary (see figure A.2). The action vector $U = (u_r, u_l)$ directly specifies the two angular wheel velocities. The following kinematics equations is used to simplify the general form of the prediction state,

$$v = \frac{u_l + u_r}{2} \ , \ \omega = \frac{u_r - u_l}{D} \tag{A.4}$$

where v, ω denote the velocity and the angular velocity of the robot respectively. From figure A.2 and equation A.4, we can calculate the change in location of the mobile robot as follows:

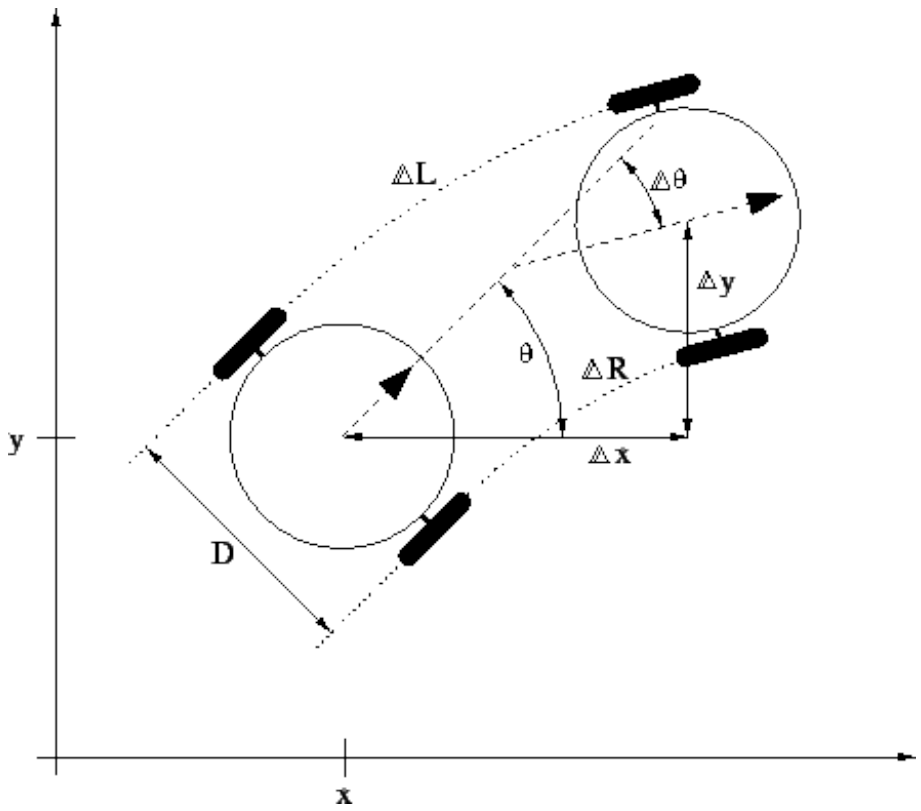


Figure A.2: Illustration of the local and global coordinate system

$$\begin{pmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{pmatrix} = \begin{pmatrix} x_v(k) \\ y_v(k) \\ \theta_v(k) \end{pmatrix} + \begin{pmatrix} \Delta T v(k) \cos\left(\theta_k + \frac{\omega}{2}\right) \\ \Delta T v(k) \sin\left(\theta_k + \frac{\omega}{2}\right) \\ \Delta T \omega(k) \end{pmatrix} \tag{A.5}$$

The form of equation A.5 represents a discrete time robot process model to be used in the prediction stage of the robot state estimator in each time step ΔT.

Equation A.5 relies on the odometry, which is the most widely used navigation method for mobile robot positioning [Borenstein 96]. It is well known that the odometry provides good short-term accuracy, is inexpensive, and allows very high sampling rates. However, the fundamental idea of odometry is the integration of incremental motion information over time, which leads inevitably to the accumulation of errors. Beacuse of that arises the need to model this uncertainty and to use vision or other sensors to assist the localization process.

The uncertainty is modelled as Gaussian variation, and can be calculated by using the standard formula for transfer of uncertainty at first order:

$$Q = \frac{\partial f_v}{\partial u} U \frac{\partial f_v}{\partial u}^T \tag{A.6}$$

where the estimated position vector f_v, and the control vector u are defined as:

$$f_v = \begin{pmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{pmatrix} \; ; \; u = \begin{pmatrix} u_r \\ u_l \end{pmatrix} \; ; \; r, l : right, left \tag{A.7}$$

$\frac{\partial f_v}{\partial u}$ is the Jacobian between fv and u, and U is the covariance matrix of u, and can be found as follows:

$$\frac{\partial f_v}{\partial u} = \begin{pmatrix} \frac{\partial x_v(k+1)}{\partial u_r} & \frac{\partial x_v(k+1)}{\partial u_l} \\ \frac{\partial y_v(k+1)}{\partial u_r} & \frac{\partial y_v(k+1)}{\partial u_l} \\ \frac{\partial \theta_v(k+1)}{\partial u_r} & \frac{\partial \theta_v(k+1)}{\partial u_l} \end{pmatrix} \; ; \; U = \begin{pmatrix} \frac{\sigma_v^2}{2} & 0 \\ 0 & \frac{2\sigma_v^2}{D^2} \end{pmatrix} \tag{A.8}$$

where σ_v is the uncertainty in the effective control input, and D is the distance between the robot wheels.

A.3.2 Landmark Model

In the context of SLAM, a landmark is a feature of the environment that can be consistently and reliably observed using the vehicle's sensors. Landmarks must be described in parametric form to allow them to be incorporated into a state model. Therefore, a robust extraction of feature points from a given scene image is used to infer the landmark locations with respect to a three dimensional world coordinate system. Given that the position of landmarks are assumed to be stationary, the landmark model of this system for all landmarks is given by:

$$\begin{bmatrix} x_{fi}(k+1) \\ y_{fi}(k+1) \\ z_{fi}(k+1) \end{bmatrix} = \begin{bmatrix} x_{fi}(k) \\ y_{fi}(k) \\ z_{fi}(k) \end{bmatrix} \tag{A.9}$$

A.3.3 Sensor Model

The vehicle shown in figure A.3 is equipped with a stereo camera sensor that takes observations of the features in the environment. At the instant k, the algorithm returns the range $z_r(k)$, bearing $z_\theta(k)$ and elevation $z_\phi(k)$ of a landmark f_i. Given the current vehicle position $x_v(k)$ and the position of an observed feature $f_i(k)$, the observation of range $z_{r_i}(k)$, bearing $z_{\theta_i}(k)$ and elevation $z_{\phi_i}(k)$ can be modelled as:

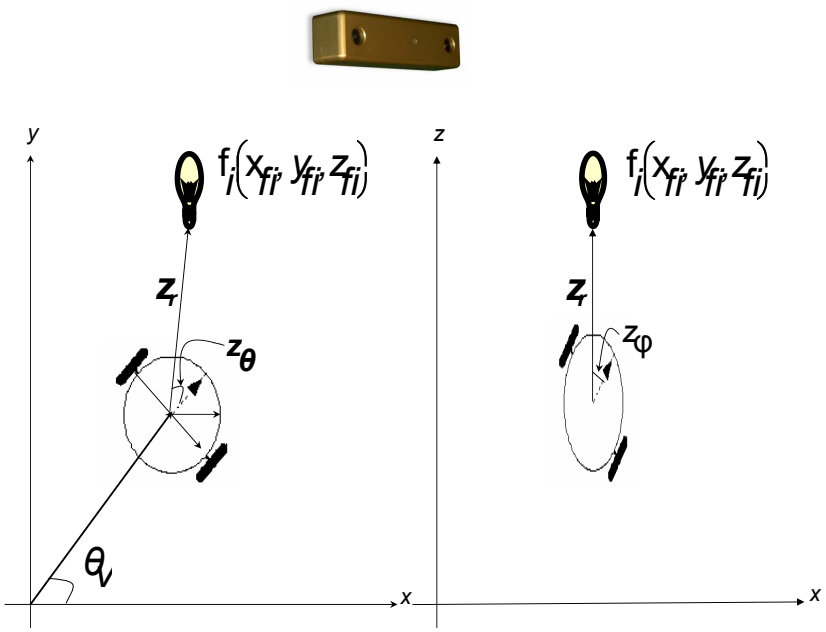


Figure A.3: Robot and observation kinematics

$$z_i(k) = \begin{bmatrix} z_{r_i}(k) \\ z_{\theta_i}(k) \\ z_{\phi_i}(k) \end{bmatrix} \tag{A.10}$$

$$= \begin{bmatrix} R_{f_i}(k) \\ \arctan\left(\frac{y_{f_i} - y_v(k)}{x_{f_i} - x_v(k)}\right) - \theta_v(k) \\ \arcsin\left(\frac{z_{f_i} - z_v(k)}{R_{f_i}(k)}\right) \end{bmatrix} + \begin{bmatrix} w_r(k) \\ w_\theta(k) \\ w_\phi(k) \end{bmatrix} \tag{A.11}$$

where $R_{f_i}(k) = \sqrt{(x_{f_i} - x_v(k))^2 + (y_{f_i} - y_v(k))^2 + (z_{f_i})^2}$ is the Euclidean distance between the robot and a landmark f_i and w_r, w_θ and w_ϕ are the noise sequences associated with the range, bearing and elevation measurements.

A.4 Estimation Process

Given the process and sensor models described in the previous sections, the localization and map building process consist of generating the best estimate for the system states given the information available to the system. This can be accomplished using a recursive, three stage procedure comprising the following steps: Moving and making a prediction, predicting a measurement and updating the state vector after a measurement.

A.4.1 Filter Initialization

We initialize the system by using the following assumptions:

- No knowledge of any scene feature.
- The global coordinate frame is aligned with the robot's starting position.

Because of that, the starting covariance matrix is set to zero. The vehicle state at the initial state is uniquely determined by its position and orientation which are set to zero as well,

$$x_v = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \; ; \; P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.12}$$

A.4.2 Moving and Making Predictions

Adopting the notation of Gelb [Gelb 94], the posterior estimate of the state at time k conditioned on the information up to time k will be written as $\hat{x}^{+}(k)$, whereas the prior estimate of the state at time k given the information up to time $k-1$, also referred to as the one-step-ahead prediction, will be written as $\hat{x}^{-}(k)$.

The prediction stage of the filter uses the model of the motion of section A.3 in order to generate an estimate of the vehicle position, $\hat{x}_v^{-}(k)$, at instant k, given the information available up to instant $k-1$. After a step of movement, a new state estimate and covariance are produced as follows:

$$\hat{x}^{-}(k) = \begin{pmatrix} f_v\left(\hat{x}_v^{+}(k-1), u(k)\right) \\ \hat{f}_i \\ \hat{f}_j \\ \hat{f}_k \\ \vdots \end{pmatrix} \tag{A.13}$$

$$P^{-}(k) = \begin{bmatrix} \frac{\partial f_v}{\partial x_v} P_{vv}^{-} \frac{\partial f_v}{\partial x_v}^T + Q(k) & \frac{\partial f_v}{\partial x_v} P_{vf_i}^{-} & \frac{\partial f_v}{\partial x_v} P_{vf_j}^{-} & \frac{\partial f_v}{\partial x_v} P_{vf_k}^{-} & \dots \\ P_{f_iv}^{-} \frac{\partial f_v}{\partial x_v}^T & P_{f_if_i}^{-} & P_{f_if_j}^{-} & P_{f_if_k}^{-} & \dots \\ P_{f_jv}^{-} \frac{\partial f_v}{\partial x_v}^T & P_{f_jf_i}^{-} & P_{f_jf_j}^{-} & P_{f_jf_k}^{-} & \dots \\ P_{f_kv}^{-} \frac{\partial f_v}{\partial x_v}^T & P_{f_kf_i}^{-} & P_{f_kf_j}^{-} & P_{f_kf_k}^{-} & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{A.14}$$

where $Q(k)$ and f_v are as defined in equation A.6 and equation A.7 respectively.

A.4.3 Predicting a Measurement

The vehicle observes the relative range, bearing and elevation between itself and features in the environment. Given the current vehicle position estimate $\hat{x}_v^- (k)$ and the estimated position of an observed feature $\hat{f}_i (k)$, the predicted observation of range $\hat{z}_{r_i}^- (k)$, bearing $\hat{z}_{\theta_i}^- (k)$ and elevation $\hat{z}_{\phi_i}^- (k)$ between the vehicle and the feature can be computed by using the observation model described in section A.3, and then applying it to the following formula:

$$\hat{z}_i^- (k) = h\left(\hat{x}^- (k)\right) \tag{A.15}$$

When a measurement of this vector is received from the vehicle's on-board sensors, the innovation covariance, which is the covariance of the difference between the predicted and measured values, is computed. To find the innovation covariance $S(k)$, the Jacobian of the observation model, $\nabla_x h (k)$, and the covariance of the observation model R_l must be calculated. When the predicted measurement is made, the innovation covariance at the prediction $\hat{x}^- (k)$ results in:

$$S (k) = \nabla_x h (k) P^- (k) \nabla_x h^T (k) + R_l \tag{A.16}$$

The calculation of the innovation covariance can be simplified by noting that each observation is only a function of the feature being observed. Therefore, the Jacobian of the observation function, $\nabla_x h (k)$ is a sparse matrix of the form

$$\nabla_x h (k) = \left[\begin{array}{ccccccc} \nabla_v h (k) & 0 & \dots & 0 & \nabla_i h (k) & 0 & \dots \end{array} \right], \tag{A.17}$$

where $\nabla_v h (k)$ is the Jacobian of the observation function h with respect to the vehicle state, whereas $\nabla_i h (k)$ is the Jacobian of the observation function h with respect to the observed feature state f_i. Rewriting equation A.16 using the sparse Jacobian results in:

$$S (k) = \frac{\partial h}{\partial x_v} P_{vv}^- \frac{\partial h}{\partial x_v}^T + 2 \frac{\partial h}{\partial x_v} P_{vf_i}^- \frac{\partial h}{\partial f_i}^T + \frac{\partial h}{\partial f_i} P_{f_i f_i}^- \frac{\partial h}{\partial f_i}^T + R_l \tag{A.18}$$

where R_l is the measurement noise covariance matrix and is represented in terms of standard deviation of the range, bearing and elevation values as:

$$R_l = \left[\begin{array}{ccc} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{array} \right] \tag{A.19}$$

A.4.4 Updating the State Vector after a Measurement

Once the observation has been associated with a particular feature in the map, the state estimate can be updated using the optimal gain matrix $W(k)$. This gain matrix provides a weighted sum of the prediction and observation and is computed using the innovation covariance, $S(k)$, and the predicted state covariance, $P^{-}(k)$. The weighting factor is proportional to $P^{-}(k)$ and inversely proportional to the innovation covariance [Smith 87b]. This knowledge is used to compute the state update $x^{+}(k)$ as well as the updated state covariance $P^{+}(k)$. Using the scalar innovation variance $S(k)$, the gain matrix $W(k)$ can then be calculated and the filter update can be performed in the usual way as follows:

$$W(k) = P^{-}(k)\,\nabla_x h^T(k)\,S^{-1}(k) \tag{A.20}$$

$$= S^{-1}(k)\begin{pmatrix} P^{-}_{vv}(k) \\ P^{-}_{f_i v}(k) \\ P^{-}_{f_j v}(k) \\ P^{-}_{f_k v}(k) \\ \vdots \end{pmatrix}\frac{\partial h_i}{\partial x_v}^T + S^{-1}(k)\begin{pmatrix} P^{-}_{v f_i}(k) \\ P^{-}_{f_i f_i}(k) \\ P^{-}_{f_j f_i}(k) \\ P^{-}_{f_k f_i}(k) \\ \vdots \end{pmatrix}\frac{\partial h_i}{\partial f_i}^T \tag{A.21}$$

$$\hat{x}^{+}(k) = \hat{x}^{-}(k) + W(k)\left(z(k) - \hat{z}^{-}(k)\right) \tag{A.22}$$

$$P^{+}(k) = P^{-}(k) - W(k)\,S(k)\,W^T(k) \tag{A.23}$$

$z(k)$ is the actual measurement of the feature obtained from the sensor, and $\hat{z}^{-}(k)$ is the prediction measurement. This update is carried out sequentially for each scalar element of the measurement.

A.4.5 Feature Initialization

When an unknown feature is observed for the first time, its estimate must be properly initialised and added to the state vector. Based on the configuration model presented on figure A.3, the initialization equation needed for adding a new feature into the state vector is as follows:

$$f_{new}(k) = \begin{pmatrix} x^{+}_{f_{new}}(k) \\ y^{+}_{f_{new}}(k) \\ z^{+}_{f_{new}}(k) \end{pmatrix} \tag{A.24}$$

$$= \begin{pmatrix} x^{-}_{v}(k) \\ y^{-}_{v}(k) \\ z^{-}_{v}(k) \end{pmatrix} + \begin{pmatrix} z_r(k)\cos(\theta_v(k) + z_\theta(k))\cos(z_\phi(k)) \\ z_r(k)\sin(\theta_v(k) + z_\theta(k))\cos(z_\phi(k)) \\ z_r(k)\sin(z_\phi(k)) \end{pmatrix} \tag{A.25}$$

It is also needed to calculate the required Jacobians based on the initialisation formula of equation A.25, for example, the Jacobian of the observation with respect to the vehicle state $\nabla_v h(k)$, and the Jacobian of the observation with respect to the observed feature $\nabla_{f_{new}} h(k)$. Suppose that the robot had observed a new feature f_{new}, then the total state vector and the covariance matrix are updated based on this information, and they result in:

$$x^{+}(k) = \begin{pmatrix} x_v(k) \\ f_i \\ f_j \\ f_k \\ f_{new} \end{pmatrix} \tag{A.26}$$

$$P^{+}(k) = \begin{bmatrix} P_{vv}(k) & P_{vf_i}(k) & P_{vf_j}(k) & P_{vf_k}(k) & P_{vf_{new}}(k) \\ P_{f_iv}(k) & P_{f_if_i}(k) & P_{f_if_j}(k) & P_{f_if_k}(k) & P_{f_if_{new}}(k) \\ P_{f_jv}(k) & P_{f_jf_i}(k) & P_{f_jf_j}(k) & P_{f_jf_k}(k) & P_{f_jf_{new}}(k) \\ P_{f_kv}(k) & P_{f_kf_i}(k) & P_{f_kf_j}(k) & P_{f_kf_k}(k) & P_{f_kf_{new}}(k) \\ P_{f_{new}v}(k) & P_{f_{new}f_i}(k) & P_{f_{new}f_j}(k) & P_{f_{new}f_k}(k) & P_{f_{new}f_{new}}(k) \end{bmatrix} \tag{A.27}$$

where the covariance between the newest feature and the vehicle and covariance between the newest feature and oldest features are defined as follows,

$$P_{f_{new}v}(k) = \frac{\partial f_{new}}{\partial x_v} P_{vv}(k)$$

$$P_{f_{new}f_i}(k) = \frac{\partial f_{new}}{\partial x_v} P_{vf_i}(k)$$

$$P_{f_{new}f_j}(k) = \frac{\partial f_{new}}{\partial x_v} P_{vf_j}(k)$$

$$P_{f_{new}f_k}(k) = \frac{\partial f_{new}}{\partial x_v} P_{vf_k}(k)$$

$$P_{f_{new}f_{new}}(k) = \frac{\partial f_{new}}{\partial x_v} P_{vv}(k) \frac{\partial f_{new}}{\partial x_v}^T + \frac{\partial f_{new}}{\partial h} R_l \frac{\partial f_{new}}{\partial h}^T$$

$$P_{f_k f_{new}}(k) = P_{f_k v}(k) \frac{\partial f_{new}}{\partial x_v}^T$$

$$P_{f_j f_{new}}(k) = P_{f_j v}(k) \frac{\partial f_{new}}{\partial x_v}^T$$

$$P_{f_i f_{new}}(k) = P_{f_i v}(k) \frac{\partial f_{new}}{\partial x_v}^T$$

$$P_{v f_{new}}(k) = P_{vv}(k) \frac{\partial f_{new}}{\partial x_v}^T$$

A.4.6 Deleting a Feature

Deleting a feature is done by removing the rows and columns from the state vector and covariance matrix which contain it. An example in a system, where the second feature f_j of three known features is deleted would be as :

$$\begin{pmatrix} x_v(k) \\ f_i \\ f_j \\ f_k \end{pmatrix} \rightarrow \begin{pmatrix} x_v(k) \\ f_i \\ f_k \end{pmatrix} \tag{A.28}$$

$$\begin{bmatrix} P_{vv}(k) & P_{vf_i}(k) & P_{vf_j}(k) & P_{vf_k}(k) \\ P_{f_i v}(k) & P_{f_i f_i}(k) & P_{f_i f_j}(k) & P_{f_i f_k}(k) \\ P_{f_j v}(k) & P_{f_j f_i}(k) & P_{f_j f_j}(k) & P_{f_j f_k}(k) \\ P_{f_k v}(k) & P_{f_k f_i}(k) & P_{f_k f_j}(k) & P_{f_k f_k}(k) \end{bmatrix} \rightarrow \begin{bmatrix} P_{vv}(k) & P_{vf_i}(k) & P_{vf_k}(k) \\ P_{f_i v}(k) & P_{f_i f_i}(k) & P_{f_i f_k}(k) \\ P_{f_k v}(k) & P_{f_k f_i}(k) & P_{f_k f_k}(k) \end{bmatrix} \tag{A.29}$$

Appendix B

Technical Data of the Used Hardware

B.1 Digital Stereo Vision Camera *bumblebee* from Point Grey Research

Figure B.1: Stereo vision camera *bumblebee*, from: [Point Grey Research 05]

Technical Properties	*bumblebee*
Image device	Two progressive Scan CCDs
Base line	$12cm$
Picture size	(640×480) Pixels
Digital interface	IEEE 1394
Frame rate	$30hz$
Dimension	approx. $160mmx40mm50mm$
Weight	approx. $375g$
Extra	Accurately precalibrated

Table B.1: Specfication data of a stereo vision camera from: [Point Grey Research 05]

B.2 Color Digital Camera Module Sony DFW-SX 900 from Sony Corporation

Figure B.2: Sony color digital camera **DFW-SX 900**, from: [Sony Corporation 05]

Technical Properties	**DFW-SX 900**
Image device	Progressive Scan CCD
Picture size	SXGA (1280 × 960) Pixel
Digital interface	IEEE 1394
Transfer rate	400 Mbps
Frame rate	7.5 frames/s
Dimension (W x H x D)	55 mm x 50 mm 110 mm
Mass	250 g

Table B.2: Specfication data of Sony **DFW-SX 900** from: [Sony Corporation 05]

B.3 Laser Scanner SICK *LMS200*

Figure B.3: Laser scanner SICK *LMS200*, from: [SICK 05]

Specification	*LMS200*
Part number	1015850
Field of view	180°
Angular resolution	$1 \ldots 0.25°$
Response time	$13 \ldots 53ms$
Resolution	$10mm$
Systematic error	$\pm 15mm$
Statistical error ($1Sigma$)	$5mm$
Scanning range	$80m$
Data Interface	RS-232, RS-422
Dimensions ($L \times W \times H$)	156 mm x 155 mm x 210 mm
Weight	4.5 kg

Table B.3: Technical data of the *LMS200* from: [SICK 05]

List of Figures

List of Tables

List of Algorithms

Bibliography

[Albers 02] S. Albers, K. Kursawe, S. Schuierer. Exploring unknown environments with obstacles. Tagungsband: *Proceedings of the 10th Symposium on Discrete Algorithms*, 2002.

[Baglietto 03] M. Baglietto, M. Paolucci, L. Scardovi, R. Zoppoli. Entropy-based environment exploration and stochastic optimal control. Tagungsband: *Proceedings of the 42nd IEEE Conference on Decision and Control*, Seiten 2938– 2941, 2003.

[Berns 05] K. Berns, K. Kleinlützum, T. Luksch, D. Schmidt. Selbstständige erstellung einer abstrakten topologiebasierten karte für die autonome exploration, 2005.

[Borenstein 96] J. Borenstein, H. R. Everett, L. Feng. *Navigating Mobile Robots*. Wellesley, 1996.

[Bourgault 02] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, H. F. Durrant-Whyte. Information based adaptive robotic exploration. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems*, 2002.

[Brooks 03] R. R. Brooks, S. S. Iyengar. *Multi-sensor fusion : fundamentals and applications with software*. Prentice Hall, 2003.

[Bruyninkx 02] H. Bruyninkx. Bayesian probability, 2002. 2005.

[Burgard 02] W. Burgard, M. Moors, F. Schneider. Collaborative exploration of unknown environments with teams of mobile robots. Tagungsband: *Proceedings of the Dagstuhl Seminar on Plan-based Control of Robotic Agents*, October 2002.

[Burgard 05] W. Burgard, M. Moors, C. Stachniss, F. Schneider. Coordinated multi-robot exploration. *Journal of IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[Burgard 96] W. Burgard, D. Fox, D. Hennig, T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. Tagungsband: *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[Cao 97] Y. Cao, A. S. Fukunaga, A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Journal of Autonomous Robots*, 4(1):7–27, March 1997.

[Chang 03] M. M. Chang, G. F. Wyeth. Achieving cooperation in a distributed multi-robot team, 2003.

[Chapron 92] M. Chapron. A new chromatic edge detector used for color image segmentation. Tagungsband: *Proceedings of the International Conference on Pattern Recognition*, August 1992.

[Comaniciu 02] D. Comaniciu, P. Meer. Mean shift: a robust approach toward feature space analysis. *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[Comaniciu 97] D. Comaniciu, P. Meer. Robust analysis of feature spaces: color image segmentation. Tagungsband: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997.

[Csorba 97] M. Csorba, J. K. Uhlmann, H. F. Durrant-Whyte. A sub-optimal algorithm for automatic map building. Tagungsband: *Proceedings of the American Control Conference*, USA, June 1997.

[Davies 05] E. R. Davies. *Machine vision : theory, algorithms, practicalities*. Elsevier, 2005.

[Davison 00] A. J. Davison, N. Kita. Active visual localisation for cooperating inspection robots. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[Davison 01] A. J. Davison, N. Kita. Sequential localisation and map-building for real-time computer vision and robotics. *Robotics and Autonomous Systems*, 36(4):171–183, 2001.

[Davison 02] A. J. Davison, D. W. Murray. Simultaneous localization and map building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, July 2002.

[Davison 98] A. J. Davison, D. W. Murray. Mobile robot localization using active vision. Tagungsband: *Proceedings of the 5th European Conference on Computer Vision*, Freiburg-Germany, 1998.

[Dissanayake 01] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. on Robotics and Automation*, 17(3), 2001.

[Durrant-Whyte 00] H. F. Durrant-Whyte, M. W. M. G. Dissanayake, P. W. Gibbens. Toward deployment of large scale simultaneous localisation and map building (slam) systems. Tagungsband: *Proceedings of the 9th International Symposium on Robotics research*, U.K., October 2000.

[Elfes 87] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Trans. on Robotics and Automation*, 3(1):249– 265, 1987.

[Fabrizi 02] E. Fabrizi, A. Saffiotti. Augmenting topology-based maps with geometric information. *Robotics and Autonomous Systems*, 40(2):91–97, 2002.

[Farinelli 04] A. Farinelli, L. Iocchi, D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(5):2015–2028, 2004.

[Feder 99] H. J. S. Feder, J. J. Leonard, C. M. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, 18(7):650–668, 1999.

[Feyrer 99] S. Feyrer, A. Zell. Detection, tracking, and pursuit of humans with an autonomous mobile robot. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 1999.

[Fox 01] D. Fox, S. Thrun, W. Burgard, F. Dellaert. Particle filters for mobile robot localization. Tagungsband: *Sequential Monte Carlo Methods in Practice*, New York, 2001.

[Fox 98] D. Fox, W. Burgard, S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 1998.

[Fox 99a] D. Fox, W. Burgard, H. Kruppa, S. Thrun. Collaborative multi-robot localization. Tagungsband: *Proceedings of the 23th German Conference on Artificial Intelligence*, 1999.

[Fox 99b] D. Fox, W. Burgard, S. Thrun. Markov localization for mobile robots in dynamic environments. *Artificial Intelligence Research*, 11:391–427, 1999.

[Franchi 07] A. Franchi, L. Freda, G. Oriolo, M. Vendittelli. A randomized strategy for cooperative robot exploration. Tagungsband: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2007)*, Rome, Italy, April 2007.

[Freda 06] L. Freda, F. Loiudice, G. Oriolo. A randomized method for integrated exploration. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 06)*, Oct 2006.

[Gelb 94] A. Gelb. *Applied optimal estimation.* MIT Press, 1994.

[Grabowski 00] R. Grabowski, L. E. Navarro-Serment, C: J. Paredis, P. K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Journal of Autonomous Robots*, 2000.

[Grewal 93] M. Grewal, A. Andrews. *Kalman Filtering: theory and practice.* Prentice-Hall, 1993.

[Guivant 01] J. E. Guivant, E. M. Nebot. Optimization of the simultaneous localization and map building algorithm for real-time implementation. *IEEE Trans. on Robotics and Automation*, 17(3):2051–2056, 2001.

[Gulrez 04] T. Gulrez, R. Al-Hmouz, Z. Chaczko. Unmanned autonomous vehicle control & slam problem in 2-d environment. Tagungsband: *Proceedings of 8th International Multitopic Conference INMIC 2004*, Dec 2004.

[Guzzoni 97] D. Guzzoni, A. Cheyer, L. Julia, K. Konolige. Many robots make short work. *AI Magazine*, 18(1):55–64, 1997.

[Haralick 92] Robert M. Haralick, Linda G. Shapiro. *Computer and robot vision.* Addison-Wesley, 1992.

[Huwedi 06] A. Huwedi, P. Steinhaus, R. Dillmann. Autonomous feature-based exploration using multi-sensors. Tagungsband: *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Heidelberg, Germany, Sept 2006.

[Jennings 99] C. Jennings, D. Murray, J. J. Little. Cooperative robot localization with vision-based mapping. Tagungsband: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 99)*, Detroit, MI, USA, Oct 1999.

[Jung 97] D. Jung, G. Cheng, A. Zelinsky. Experiments in realising cooperation between autonomous mobile robots. Tagungsband: *Proceedings of International Symposium on Experimental Robotics (ISER'97)*, Barcelona, June 1997.

[Kaelbling 96] L. P. Kaelbling, A. R. Cassandra, J. A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'96)*, Osaka, Japan, Nov 1996.

[Knight 01] J. Knight, A. Davison, I. Reid. Towards constant time slam using postponement. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 01)*, Maui, HI, USA, Oct 2001.

[Kuipers 91] B. Kuipers, Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[Larsen 99a] T. D. Larsen, N. A. Anderson, O. Ravn. A new approach for kalman filtering on mobile robots in thepresence of uncertainties. Tagungsband: *Proceedings of the IEEE International Conference on Control Applications*, Kohala Coast, HI, USA, Aug 1999.

[Larsen 99b] T. D. Larsen, K. L. Hansen, N. A. Anderson, O. Ravn. Design of kalman filters for mobile robots; evaluation of thekinematic and odometric approach. Tagungsband: *Proceedings of the IEEE International Conference on Control Applications*, Kohala Coast, HI, USA, Aug 1999.

[Latombe 91] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[Lee 03] D. Lee. *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Mobile Robot*. Cambridge Univeristy Press, 2003.

[Lee 04] J. Lee, N. L. Doh, W. K. Chung, B. You, Y. I. Youm. Door detection algorithm of mobile robot in hallway using pc-camera. Tagungsband: *Proceedings of the 21th International Symposium on Automation and Robotics*, July 2004.

[Lee 97] D. Lee, M. Recce. Quantitative evaluation of the exploration strategies of a mobile robot. *International Journal of Robotics Research*, 1997.

[Leonard 91] J. J. Leonard, H. F. Durrant-Whyte. Simultaneous map building and localisation for an autonomous mobile robot. Tagungsband: *Proceedings of the Workshop on Intelligent Robots and Systems (IROS)*, Nov 1991.

[Leonard 99] J. J. Leonard, H. J. S. Feder. Decoupled stochastic mapping. *IEEE Transactions on Robotics and Automation*, December 1999.

[Little 98] J. J. Little, J. Lu, D. R. Murray. Selecting stable image features for robot localization using stereo. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 98)*, Victoria, BC, Canada, Oct 1998.

[Liu 94] J. Liu, Y. Yang. Multiresolution color image segmentation. *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):689–700, July 1994.

[Makarenko 04] A. A. Makarenko, S. B. Williams, F. Bourgault, H. F. Durrant-Whyte. An experiment in integrated exploration. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 04)*, Sept 2004.

[Martinelli 02] A. Martinelli. Evaluating the odometry error of a mobile robot. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems*, October 2002.

[Martinez-Cantin 07] R. Martinez-Cantin, N. de Freitas, J. A. Castellanos. Multi-robot marginal-slam. Tagungsband: *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI), Workshop on Multirobotic Systems for Societal Applications*, Hyderabad, India, January 2007.

[Mataric 92] M. J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Trans. on Robotics and Automation*, 8(3):304–312, 1992.

[Mataric 95] M. J. Mataric, M. Nilsson, K. T. Simsarin. Cooperative multi-robot box-pushing. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'95)*, Pittsburgh, PA, USA, Aug 1995.

[Meier 05] D. Meier, C. Stachniss, W. Burgard. Coordinating multiple robots during exploration under communication with limited bandwidth. Tagungsband: *Proceedings of the European Conference on Mobile Robots (ECMR)*, Ancona - Italy, 2005.

[Mendle 71] J. Mendle. Computational requirements of a discrete kalman filter. *IEEE Transactions on Automatic Control*, 16(6):748–758, Dec 1971.

[Miro 05] J. V. Miro, G. Dissanayake, W. Zhou. Vision-based slam using natural features in indoor environments. Tagungsband: *Proceedings of IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Dec 2005.

[Montemerlo 02] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. Tagungsband: *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

[Montemerlo 03] M. Montemerlo, S. Thrun. Simultaneous localization and mapping with unknown data association using fastslam. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Sept 2003.

[Moorehead 93] S. J. Moorehead, R. Simmons, W. L. Whittaker. Autonomous exploration using multiple sources of information. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1993.

[Moravec 88] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[Moravec 89] H. Moravec. Certainty grids for sensor fusion in mobile robots. *Sensor Devices and Systems for Robotics*, Seiten 243–276, 1989.

[Moutarlier 89] P. Moutarlier, R. Chatila. Stochastic multi-sensory data fusion for mobile robot location and environment modelling. Tagungsband: *Proceedings of the 5th International Symposium on Robotics Research*, Tokyo, 1989.

[Murphy 99] K. Murphy. Bayesian map learning in dynamic environments, 1999.

[Murray 97] D. Murray, C. Jennings. Stereo vision based mapping and navigation for mobile robots. Tagungsband: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 97)*, April 1997.

[Nevado 04] M. N. Nevado, J. G. Garcia-Bermejo, E. Z. Casanova. Obtaining 3d models of indoor environments with a mobile robot by estimating local surface directions. *Robotics and Autonomous Systems*, 48:131–143, 2004.

[Newman 01] P. M. Newman, H. F. Durrant-Whyte. A new solution to the simultaneous and map building (slam) problem. Tagungsband: *Proceedings of the SPIE Conference*, Boston, USA, November 2001.

[Newman 02] P. M. Newman, J. W. Fenwick, J. J. Leonard. Cooperative concurrent mapping and localization. Tagungsband: *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, Washington, USA, May 2002.

[Newman 03] P. Newman, M. Bosse, J. Leonard. Autonomous feature-based exploration. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Sept 2003.

[Palmer 99] S. E. Palmer. *Vision Science: Photons to Phenomenology.* MIT press, 1999.

[Pandey 07] A. K. Pandey, K. M. Krishna, M. Nath. Feature based occupancy grid maps for sonar based safe-mapping. Tagungsband: *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, Hyderabad, India, Jan 2007.

[Parker 98] L. E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Trans. on Robotics and Automation*, 14(2):220–240, 1998.

[Plataniotis 01] Konstantinos N. Plataniotis, Anastasios N. Venetsanopoulos. *Color image processing and applications.* Springer, 2001.

[Point Grey Research 05] Point Grey Research. Stereo bumblebee, 2005. Version from 14.02.2005.

[Rano 01] I. Rano, E. Lazkano, I. Zarautz, I. Monasterio, B. Sierra. Mobile robot navigation using color image segmentation, 2001.

[Rekleitis 00] I. M. Rekleitis, G. Dudek, E. E. Milios. Multi-robot collaboration for robust exploration. Tagungsband: *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, 2000.

[Rekleitis 01] I. Rekleitis, R. Sim, G. Dudek, E. Milios. Local and global localization for mobile robots using visuallandmarks. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'01)*, Maui, HI, USA, Oct 2001.

[Rekleitis 04] I. Rekleitis, V. Lee-Shue, A. P. New, H. Choset. Limited communication, multi-robot team based coverage. Tagungsband: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2004)*, New Orleans, USA, April 2004.

[Rencken 94] W. D. Rencken. Autonomous sonar navigation in indoor, unknown and unstructuredenvironments. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems*, Sept 1994.

[Rocha 05] R. Rocha, J. Dias, A. Carvalho. Cooperative multi-robot systems: A study of vision-based 3d mapping using information theory. *Robotics and Autonomous Systems*, 53:282–311, 2005.

[Rous 05] M. Rous, H. Lüpschen, K. F. Kraiss. Vision-based indoor scene analysis for natural landmark detection. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, April 2005.

[Roy 99] N. Roy, W. Burgard, D. Fox, S. Thrun. Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. Tagungsband: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 1999)*, USA, Oct 1999.

[Russell 03] Stuart J. Russell, Peter Norvig. *Artificial intelligence : a modern approach*. Prentice Hall, 2003.

[Sack 03] D. Sack, W. Burgard. A comparison of methods for line extraction from range data. Tagungsband: *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2003.

[Scholl 00a] K. U. Scholl. Modular controller architecture (MCA) version 2, 2000.

[Scholl 00b] K. U. Scholl, V. Kepplin, J. Albiez, R. Dillmann. Developing robot prototypes with an expandable modular controller architecture. Tagungsband: *Proceedings of the International Conference on Intelligent Autonomous Systems*, Venice, 2000.

[Scholl 01] K. U. Scholl, J. Albiez, B. Gassmann. MCA – An expandable modular controller architecture. Tagungsband: *Proceedings of the 3rd Real-Time Linux Workshop*, Milano, Italy, 2001.

[Schultz 99] A. Schultz, W. Adams, B. Yamauchi. Integrating exploration, localization, navigation and planning with a common representation. *Journal of Autonomous Robots*, 3:293–308, 1999.

[Se 01] S. Se, D. Loweand J. Little. Local and global localization for mobile robots using visuallandmarks. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'01)*, Oct 2001.

[Shapiro 94] Linda G. Shapiro, George C. Stockman. *Computer vision*. Prentice Hall, 1994.

[SICK 05] SICK. SICK LMS 200 Product Information, 2005. Version from 12.10.2005.

[Siegwart 04] R. Siegwart, I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[Sim 01] I. Rekleitisand R. Sim, G. Dudek, E. Milios. Collaborative exploration for the construction of visual maps. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems*, 2001.

[Sim 06] R. Sim, J. J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 06)*, Oct 2006.

[Simmons 00] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, H. Younes. Coordination for multi-robot exploration and mapping. Tagungsband: *Proceedings of the AAAI National Conference on Artificial Intelligence*, Austin, TX, 2000.

[Singh 93] K. Singh, K. Fujimura. Maping by cooperatively mobile robots. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1993.

[Singhal 97] A. Singhal. Issues in autonomous mobile robot navigation, 1997.

[Smith 01] Penelope Probert Smith. *Active sensors for local planning in mobile robotics*. World Scientific, 2001.

[Smith 87a] R. C. Smith, P. Cheesman. On the representation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1987.

[Smith 87b] R. C. Smith, R. Self, P. Cheesman. Estimating uncertain spatial relationships in robotics. Tagungsband: *Proceedings of the International Conference on Robotics and Automation*, 1987.

[Smith 97] S. M. Smith, J. M. Brady. Susan—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[Sojka 03] S. Sojka. A new approach to detecting the corners in digital images. Tagungsband: *Proceedings of the International Conference on Imagerocessing (ICIP)*, 2003.

[Sonka 03] Milan Sonka, Vaclav Hlavac, Roger Boyle. *Image processing, analysis, and machine vision*. PWS Publ, 2003.

[Sony Corporation 05] Sony Corporation. Sony Digital Color Camera DFW-SX 900, 2005. Version from 28.01.2005.

[Stachniss 06] Cyrill Stachniss. *Exploration and mapping with mobile robots*. Dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau, 2006.

[Steinhaus 03] P. Steinhaus, R. Dillmann. Aufbau und modellierung des rosi scanners zur 3d-tiefenbildakquisition, 2003.

[Surmann 03] H. Surmann, A. Nüchter, J. Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45:181–198, 2003.

[Takezawa 04] S. Takezawa, D. C. Herath, D. G. Dissanayake. Slam in indoor environments with stereo vision. *IEEE Transactions on Robotics and Automation*, 2:1866–1871, Setember 2004.

[Tan 97] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. *Machine Learning*, Seiten 487–494, 1997.

[Taylor 05] T. Taylor, S. Geva, W. W. Boles. Directed exploration using a modified distance transform. Tagungsband: *Proceedings of the International Conference on Digital Image Computing*, Dec 2005.

[Taylor 06] Geoffrey Taylor, Lindsay Kleeman. *Visual perception and robotic manipulation : 3D object recognition, tracking and hand-eye coordination*. Springer, 2006.

[Taylor 93] C. J. Taylor, D. J. Kriegman. Exploration strategies for mobile robots. *IEEE Transactions on Robotics and Automation*, 2:248–253, May 1993.

[Thrun 00] S. Thrun, W. Burgard, D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000.

[Thrun 04] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, H. Durrant-Whyte. Simultaneous localizations and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23:693–716, 2004.

[Thrun 06] S. Thrun, W. Burgard, D. Fox. *Probabilistic Robotics*. The MIT Press, 2006.

[Thrun 96] S. Thrun, A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. Tagungsband: *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, 1996.

[Thrun 98a] S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1):41–76, 1998.

[Thrun 98b] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[Thrun 98c] S. Thrun, D. Fox, W. Burgard. a probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[Thrun 98d] S. Thrun, D. Fox, W. Burgard. Probabilistic mapping of an environment by a mobile robot. *IEEE Transactions on Robotics and Automation*, 2:1546–1551, May 1998.

[Tovary 06] B. Tovary, L. Munoz-Gomez, R. Murrieta-Cidz, M. Alencastre-Mirandaz, R. Monroyz, S. Hutchinsony. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4):314–331, 2006.

[Vidal 02] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation*, Oct 2002.

[Vonderhardt 96] H. J. Vonderhardt, D. Wolf, R. Husson. The dead reckoning localization system of the wheeled mobile robot romane. Tagungsband: *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Dec 1996.

[Walther 06] M. Walther, P. Steinhaus, R. Dillmann. A foveal 3d laser scanner integrating texture into range data. Tagungsband: *Proceedings of the 9th International Conference on Intelligent Autonomous Systems*, April 2006.

[Wang 88] C. M. Wang. Location estimation and uncertainty analysis for mobile robots. Tagungsband: *Proceedings of IEEE International Conference on Robotics and Automation*, April 1988.

[Wasik 02] Z. Wasik, A. Saffiotti. Robust color segmentation for the robocup domain. Tagungsband: *Proceedings of the 2002 IEEE 16th International Conference on Pattern Recognition*, Oct 2002.

[Wawerla 02] J. Wawerla, G. S. Sukhatme, M. J. Mataric. Collective construction with multiple robots. Tagungsband: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'02)*, 2002.

[Weiss 00] Gerhard Weiss. *Multiagent systems : a modern approach to distributed artificial intelligence*. MIT Press, 2000.

[Welch 01] Greg Welch, Gary Bishop. An introduction to the kalman filter, 2001. Course by SIGGRAPH 2001.

[Williams 02a] S. B. Williams, G. Dissanayake, H. F. Durrant-Whyte. An efficient approach to the simultaneous localisation. Tagungsband: *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, Washington, Usa, May 2002.

[Williams 02b] S. B. Williams, G. Dissanayake, H. F. Durrant-Whyte. Towards multi-vehicle simultaneous localisation and mapping. Tagungsband: *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, Washington, USA, May 2002.

[Yamauchi 97] Brian Yamauchi. A frontier-based approach for autonomous exploration. Tagungsband: *Proceedings of the International Conference on Computational Intelligence in Robotics and Automation*, USA, 1997.

[Yamauchi 98] Brian Yamauchi. Frontier-based exploration using multiple robots. Tagungsband: *Proceedings of second International Conference on Autonomous Agents*, USA, 1998.

[Zelinsky 92] A. Zelinsky. A mobile robot navigation exploration algorithm. Tagungsband: *Proceedings of the IEEE International Conference on Robotics and Automation*, December 1992.

[Zhang 03] H. Zhang, K. Yuan, J. Liu. A fast and robust vision system for autonomous mobile robots. Tagungsband: *Proceedings of the International Conference on Robotics, Intelligent Systems and Signal Processing*, 2003.